(12) LEVEL II

RADC-TR-81-190, Vol I (of two)
Final Technical Report
September 1981

ADVANCED WEAPON SYSTEM (AWS) SENSOR PREDICTION TECHNIQUES STUDY

General Electric Company

C. F. R. Weiman

ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss Air Force Base, New York 13441

"Original contains color plates: All DTIC reproductions will be in black and white"

This report has been reviewed by the RADC Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

Because of the size of this document, it has been divided into two volumes. Volume I contains Sections 1 through 5. Volume II contains Appendix IV.

RADC-TR-81-190, Volume I (of two) has been reviewed and is approved for publication.

APPROVED: Robert H. La Salle

ROBERT H. LA SALLE
Project Engineer

APPROVED:

OWEN R. LAWTER, Colonel, USAF
Chief, Intelligence and Reconnaissance Division

FOR THE COMMANDER: John P. Huss

JOHN P. HUSS
Acting Chief, Plans Office

If your address has changed or if you wish to be removed from the RADC mailing list, or if the addressee is no longer employed by your organization, please notify RADC (IRRE) Griffiss AFB NY 13441. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document requires that it be returned.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER RADC-TR-81-190, Vol I (of two)	2. GOVT ACCESSION NO. AD-A108 792	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) ADVANCED WEAPON SYSTEM (AWS) SENSOR PREDICTION TECHNIQUES STUDY		5. TYPE OF REPORT & PERIOD COVERED Final Technical Report 14 Feb 80 - 11 May 81 6. PERFORMING ORG. REPORT NUMBER N/A
7. AUTHOR(s) C.F.R. Weiman		8. CONTRACT OR GRANT NUMBER(s) F30602-80-C-0082
9. PERFORMING ORGANIZATION NAME AND ADDRESS General Electric Company P O Box 2500 Daytona Beach FL 32015		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 62702F 45941718
11. CONTROLLING OFFICE NAME AND ADDRESS Rome Air Development Center (IRRP) Griffiss AFB NY 13441		12. REPORT DATE September 1981 13. NUMBER OF PAGES 244
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Same		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Same

18. SUPPLEMENTARY NOTES

RADC Project Engineer: Robert H. LaSalle (IRRP)

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

AWS, Digital Data Base, Sensor Simulation, Digital Landmass Simulator, Image Prediction, FLIR, Computer Graphics, CIG

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This study investigated, identified, and developed technology necessary to support efficient computer simulation of Advanced Weapon Systems (AWS) sensor images associated with low altitude flight. A main objective of the study was to delineate approaches to the solution of data base storage and processing problems characteristic of low altitude flight sensor image simulation. The tasks carried out to achieve this objective included analysis of scene content requirements, a survey of relevant computer

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

graphics and image processing technology, and evaluation of experimental image generation algorithms carried out by General Electric. The functional description of a research laboratory system for sensor display simulation is presented.

One of the most significant conclusions drawn from the analysis of technology and experimentation is that grid data base representation and display may dramatically improve image quality and reduce cost of sensor simulations vis-a-vis polygon data base representation in the near future. This is due to the recent advances in VLSI architecture and the discovery of new parallel algorithms in computer graphics.

TABLE OF CONTENTS - VOLUME 1

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS - VOLUME 2

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS (Continued)

LIST OF ILLUSTRATIONS (Continued)

LIST OF TABLES

SUMMARY

1.0 TECHNICAL PROBLEM

As flight missions are conducted at increasingly lower altitudes and as the resolution of imaging sensors continues to improve, the content of scenes generated by devices used for simulation and/or prediction purposes must become accordingly higher if visual cues vital to navigation and targeting are to be successfully perceived. Unfortunately, this higher scene complexity usually translates to substantially increased cost and other penalties in systems which synthesize such scenes. This is especially true if attempts are made to replicate "real" images in every detail—including both spatial and temporal aspects. The fundamental issue, then, is one of scene richness versus processing cost.

A potential resolution of this issue lies in reducing the content of visual/sensor images to "essential information" or, conversely, deleting those scene features which do not contribute significantly to the mission-specific task(s) being performed. This approach to the problem has been the subject of a number of research efforts which thus far appear to be producing some encouraging results.

Alternatively, recent advances in techniques and devices employed in the generation of visual/sensor images hold promise for reducing costs associated with processing and storage. The overall objective of the AWS Sensor Prediction Techniques Study, then, was to investigate, identify, and develop technology necessary to support efficient computer image generation (CIG) processes employed in the synthesis of low-altitude, sensor-related scenes. Having pioneered the development of CIG and maintained a position of leadership in this field for many years, General Electric placed heavy emphasis on the processing efficiency aspects of image generation in performing this study.

2.0 APPROACH/METHODOLOGY

The study was organized into four principal tasks which were undertaken in sequential fashion. The initial effort was largely one of identifying, accumulating, and digesting literature estimated to have relevance to the subject of visual and sensor scene representation for low-altitude applications. Also included were discussions with the RADC Technical Officer for the study, and interviews with General Electric personnel having past and current experience in the field. This activity provided the necessary background and enabled the definition of generalized requirements upon which subsequent study tasks could be based.

The second task encompassed an extensive and thorough review of technology judged to be supportive to the objectives of the study. The material that was researched generally fell within the following topics or categories:

- Perceptual psychology in visual training.
- Artificial vision and image processing.
- FLIR, LLLTV, microwave, and other visual sensor displays.
- Computer graphics.

- Computer image generation and simulation.
- Grid data base display and generation.

The basic intent of this task was to identify technology and/or techniques which could be applied to increase scene generation efficiency.

On the basis of the results of the second task, the techniques evaluation phase was begun. This involved the selection of candidate scene generation techniques for further evaluation via analytical and/or experimental means, acquisition of appropriate test data bases, generation of sample test scenes illustrative of the respective techniques employed, and subjective evaluation of those scenes.

Performance of the fourth and final task involved the preliminary design of a laboratory non-realtime (static) image synthesis system intended to be used in the development and evaluation of techniques, algorithms, and/or methodologies for generating and displaying visual and sensor images suitable for low-altitude, close approach applications.

3.0 TECHNICAL RESULTS

3.1 BACKGROUND INVESTIGATION

The initial study activity was that of determining the required/desired content or range of attributes of visual and sensor scenes to be synthesized in as quantitative a manner as possible. Some of the study findings were that: 1) scene requirements are heavily dependent on mission application, and somewhat less dependent on the type of sensor being used, 2) the "stylizing" of scenes involving departures from the mathematical reality of the world has been proven to be effective in many cases, and 3) the "realism versus effectiveness" issue will continue to be a matter of subjective judgment and controversy. The overall conclusion reached was that while much research has been (or is being) performed and reported on in this area, workable systematic methodologies for quantitatively specifying scene content parameters do not as yet seem to be available.

A logical starting point for the development of a data base for low-altitude operation is the topographic map or chart. Since a pilot or navigator typically must correlate significant map features with those perceived either out-the-window or via imaging sensor displays, whatever is contained in the map should also be resident in the data base. The process of defining scene content and attendant data base requirements is illustrated by example analysis.

An important consideration in quantifying scene complexity in terms of number of "edges" required to depict terrain, culture, and three-dimensional objects is the relationship between aircraft altitude and the potentially viewable area of the Earth's surface. The limiting factor when flying extremely low over flat ground and viewing small objects on the ground is the curvature of the Earth. Other considerations are atmospheric attenuation and display raster resolution, both of which limit the range (usually only a few miles) at which objects can be detected and/or recognized.

An effective and often-used method for reducing scene complexity is variable level-of-detail (LOD), since there is no need to store and display distant terrain at the same

level of detail as nearby terrain. One approach is to partition the environment into "blocks"; scene components that are not visible to the pilot, or which are too detailed to be perceived, will not be displayed. Another more common approach is to employ a "triangulation" technique which involves a different triangulation for each LOD. Example analyses to determine processing and storage requirements are presented for each approach.

3.2 TECHNOLOGY SURVEY

Visual and sensor scene simulation or prediction is a specialized application of computer graphics involving geometric abstractions of a real world environment modeled for the purpose of generating graphics images. Computer graphics provides a rich source of tools for representing, storing, and rendering such images; however, trade-offs between algorithms, data structures, and hardware must be made in the selection of an efficient scene synthesis system for specific applications.

The critical computation expense in sensor and visual scene generation arises from the extreme complexity of calculating hidden parts occlusion. This problem looms especially large in depicting images of detailed terrain in low level flight. The most efficient approach to solving this problem is the list priority method, in which occlusion relationships are precalculated. This approach was developed by General Electric in the early 1960's and has been successfully implemented in CIG hardware by General Electric and by several other companies. While many alternative techniques have been developed in industrial and academic environments for producing extremely rich artificial images, most of these exact extreme penalties in terms of processing speed and storage. The use of CIG in visual flight simulation has provided a crucible for weeding out inefficient processing from the gamut of algorithms found in non-realtime computer graphics. The surviving algorithms and system designs are often worthy candidates for software emulation in sensor simulation research.

Several CIG approaches embodied in visual systems produced by General Electric and other companies were reviewed and synopsized in terms of their architectures and respective unique characteristics.

An extensive review of image processing techniques applicable to extraction of meaningful information from extremely complex geometric inputs (images characterized as regular mosaics of intensity measures) was conducted. The objective in applying such techniques is the potential for effecting drastic compressions of geometric information in representing scenes. While many of the techniques reviewed are valuable for aiding human performance in image interpretation, most have failed to perform in a totally automatic fashion. It is estimated that image enhancement techniques could greatly improve the pilot's view of the world via sensor displays; experimentation with such techniques using a laboratory sensor simulation system may prove fruitful.

3.3 TECHNIQUES EVALUATION

The techniques evaluation phase of the study considered two fundamentally different approaches to scene generation based on two radically different scene data base structures. The first of these, which is employed almost universally in current CIG-based visual and electro-optical sensor scene generation systems, is the polygon approach. In this approach, terrain is represented quite efficiently by joining surface-specific points to form a network of triangular faces which in turn compose a piece-

wise planar approximation of the Earth's relief. The color and texture necessary to represent culture and/or planimetry (e.g., lakes, forests, swamps, fields, etc.) of the terrain can be incorporated in a number of ways. For example, culture/planimetric features defined by polygons and corresponding material descriptors (as, for example, defined in the Defense Mapping Agency Digital Landmass Data Base) can be intersected with the triangular terrain network, or a technique known as Voronoi tessellation may be applied. The latter involves defining culture polygons by "spreading" from center points until neighbors meet. Three-dimensional objects (e.g., houses, trees, vehicles, etc.) are placed on the terrain using classical cartesian coordinate geometry. A number of schemes are available for representing polyhedra, the choice of which is dependent on application, processing/storage efficiency, and display technique.

In scene generation systems employing the polygon approach, a very effective and efficient technique for providing enhanced perspective cues involves application of synthetically generated texture patterns. These require little storage, and can be mapped onto polygon surfaces at some increase in the display processing cost. The parameters of the mathematical equations used to generate texture can be adjusted to yield patterns which match the visual characteristics of plowed fields, forests, ocean waves, industrial and urban areas, and other types of culture.

Software systems capable of generating static visual scenes from polygon data bases, and which run on general-purpose minicomputers, are available from a number of sources. General Electric has developed static scene generation software packages for use in evaluating new algorithms intended for applications in realtime CIG systems prior to committing to hardware implementation. These are capable of generating a single frame in full color within a few minutes to over an hour, depending on scene richness, processing complexity, and the characteristics of the scene generation system employed.

While the polygon approach has been used successfully in many applications, the higher scene content required for low-altitude flight introduces data storage and processing problems which are tending to offset the efficiency of that technique. Terrain visual content per unit area is achieved by adding texture to polygons and increasing the "fineness" of polygonal representation of rough terrain, resulting in very large data bases which must incorporate an interlocking data structure in which the level of detail is decreased at extreme ranges to prevent system overload. Such overloads are often caused by large numbers of terrain polygons overlapping relatively few raster display lines near the horizon.

An alternative scene synthesis approach addressed in this study is that which entails the direct display of grid data bases. Whereas hardware processing costs per unit of image resolution increase with polygon-based methods, decreases may be experienced with grid-based methods. This is due to the fact that recent advances in parallel processing algorithm design and VLSI (very large scale integration) are tipping the balance in favor of the latter method; many historical disadvantages of grid techniques are now diminishing.

Most grid rendering algorithms are variants of "ray tracing", wherein each pixel is assumed to be an image of a ray projected from the viewpoint through the terrain surface. Although difficulties arise from such aspects as calculation of hidden parts, ad hoc handling of sampling mismatches due to perspective effects, and the lack of geometrically valid interpolation between elevation points, a number of effective grid rendering approaches have recently been described in the literature. Moreover,

General Electric has just completed the successful development of a proprietary hardware scene generation and display system which produces images in a matter of seconds directly from grid data bases.

Most existing algorithms for grid rendering embody the following general approach:

1) **Surface normal vectors are computed for each grid point by using neighboring elevations to approximate local surface slope.**
2) **These normal vectors are used to generate surface shading at each point by assuming surface brightness is a function of the angle between the normal and an illumination vector.**
3) **The result, which resembles a shaded relief map, is used to generate a perspective image by ray-tracing terrain points through display screen pixels to a single viewpoint.**

While the surface normal and surface shading computations are quite straightforward, implementation of the perspective display computations requires careful consideration if memory access and processing are to be optimized. One efficient azimuth ray tracing algorithm involves dividing the display screen into columns of pixels which correspond to equal azimuth sectors cutting through the data base. Each screen column is swept from bottom to top so that screen raster increments correspond to successive range increments in the data base. The images which result exhibit high realism, and various enhancements such as hazing or texturing may be introduced. Unfortunately, azimuth ray tracing algorithms generate undesirable artifacts at very long ranges and very short ranges due to oversampling and undersampling of the data grid by pixel projections, the latter being a serious problem for low-level sensor imagery. Although these effects can be overcome by applying oversampling and smoothing techniques, the random accessing of grid data memory limits parallel processing efficiency.

A new algorithm has been devised which has promise for overcoming the problems just mentioned. This algorithm accesses rows and columns of data independently. It completely decouples depth-related calculations (haze, occulting, and perspective) from scanline related calculations (azimuth ray images). Such properties encourage parallelism in the design of compact, inexpensive hardware systems for generating realistic, geometrically valid perspective images of terrain directly from Defense Mapping Agency (DMA) data. These properties also provide for straightforward enhancement of close range scenes by elevation interpolation and the interleaving of models or special features during either data definition or display generation. The grid data structure is ideal for fractal interpolation to generate realistic terrain images at close ranges without storing the corresponding elevation data.

The algorithm is designed to render perspectively correct color displays of grid data bases. Anti-aliasing, valid occlusion of invisible features (priority) and distance fading (haze) calculations are inherent in the algorithm. Computational organization exploits highly parallel hardware efficiency by making row and column calculations simple, local and independent. Trigonometric functions are avoided entirely and access to the grid data base is by row and column rather than random. The efficiency of the algorithm depends on the fact that the grid coordinate system is aligned with the viewray through the viewpoint perpendicular to the viewscreen. This results in constant range for all grid points in a row parallel to the screen, which greatly simplifies data access and computation.

Visual artifacts occurring at short and long ranges when using the conventional ray tracing algorithm are avoided by resampling the data rows to generate new elevation

posts whose spacing is precisely that of viewrays through laterally adjacent pixels. Since the viewscreen is aligned with data base coordinates, the viewray intersections with a row of elevation posts are equally spaced at any particular range. This property permits the use of a fixed resampling scheme for all points in any row of constant range. The computational advantages are that data may be accessed in stream fashion rather than randomly, and the resampling parameters need only be updated once per data row.

A variety of interpolation schemes may be applied in the near-range zone, the choice being a function of the criteria to be satisfied. For example, since "roughness" may afford better visual cues in simulation applications, random processes yielding surfaces and curves known as "fractals" may be used to generate strikingly realistic artificial terrain. Although mathematically complex, computed values of fractals can be stored in a lookup table. Further, they have a property which assures unlimited continuous level-of-detail change down to microscopic levels using a single lookup table while preserving visual stability of the synthesized terrain. Fractal methods are particularly suitable for grid data bases owing to geometric coherence of the data structure.

For the long range case, the aliasing artifacts of ray tracing methods caused by undersampling are eliminated in the new algorithm by means of a horizontal resampling technique.

3.4 SYSTEM DESIGN

The fourth and final phase of the study resulted in the specification, preliminary design, and description of an Advanced Image Synthesis System composed of a combination of hardware and software resources to be used in the development and evaluation of techniques, algorithms, and/or methodologies for generating and displaying visual and sensor scenes. Desirable overall system performance characteristics were to include but not be limited to the following:

- High quality/fidelity scene generation and display.
- Reasonably high scene generation speed (optimized balance between time and cost).
- Reasonably simple operation, good human interface provisions.
- Large storage capacity.
- Flexibility for readily accommodating new/revised algorithms, techniques, procedures, etc.
- Modular growth/expansion capability (hardware and software).
- Capability to support efficient data base generation/modification.
- Commercial, off-the-shelf hardware.

Using these characteristics as general guidelines and considering the performance of available devices and software packages, specific performance requirements were defined for the system. Also, the two principal system tasks of data base development and data base display were detailed—in many cases down to the individual command level and including descriptions of typical user interactions with the system.

The recommended Advanced Image Synthesis System is composed of off-the-shelf equipment configured around a Digital Equipment Corporation VAX 11/780 computer with 512K to 1024K bytes of core storage. This computer is recommended owing to its fast execution speed and 32-bit word size which make it ideal for image processing applications. Also, the reliability and ease of use of the VAX are such that it is becoming the industry standard.

Major subsystems are the computer and graphic work station, comprised as follows:

Computer Subsystem

- **1 - Medium-size, 32-bit computer system (with 512 kilobytes to 1 megabyte of core storage)**
- **1 - Array processor (optional)**
- **1 - Line printer**
- **2 - Disk drives (one 67 mb and one 300 mb)**
- **2 - Nine-track magnetic tape units**
- **1 - System console**

Up to 3 interactive terminals

Graphic Workstation

- **1 - Digitizing tablet and stylus**
- **1 - Color CRT monitor (with 3 to 6 channels of refresh memory)**
- **1 - Black and white CRT monitor (with refresh memory)**
- **1 - Calligraphic display and light pen**
- **1 - Polaroid film image recorder (optional)**
- **1 - Magnetic video disk (optional)**
- **1 - Color image scanner/digitizer (optional)**

4.0 CONCLUSIONS AND RECOMMENDATIONS

The analysis and evaluation of techniques carried out in this study led to the conclusions and recommendations described below.

Scene content analysis revealed that scene data bases should contain all features present on large scale flight maps because the cartographic process specifically takes human factors aspects of pilot tasks into account. Hence, features depicted on maps are highly relevant to what a pilot will pay attention to during a mission. Field-of-view visual density analysis revealed that additional models and texture must be added to the map information to provide sufficient visual cues for low level flight.

A review of the literature in computer graphics, CIG, and image processing was conducted from the point of view of special aspects of sensor scene simulation. The conclusion was drawn that hidden surface algorithm efficiency is of great importance for sensor scene simulation due to the large number of terrain faces which lie in line with the horizon for low altitude views. Algorithms which take advantage of the fact that terrain topology represents a surface network greatly reduce the complexity of hidden surface calculations by avoiding complex vector operations. Raster scan displays are recommended over calligraphic displays because of the latter's inability to adequately depict shading and color. While scanline algorithms for display generation have been popular in the past, random-access frame buffer algorithms may now be more efficient for scan conversion because of decreasing cost of memory.

A laboratory system for research in sensor scene simulation ought to be based on a 32-bit medium sized computer with graphic peripherals, and about a megabyte of core. Sixteen-bit minicomputers are too slow, small and inaccurate for sensor scene simulation. Software for display generation should be written in higher level languages and structured according to good software engineering principles.

One of the most significant conclusions of this study is that the direct display of grid data bases will replace current polygon data base methods in the near future because of recent advances in VLSI and parallel algorithm design. Images of markedly superior quality over polygon approaches could be generated efficiently from grid data bases using a new algorithm developed during this study, but not yet implemented. It is strongly recommended that grid data base display methods be pursued in further studies.

SECTION 1

INTRODUCTION

1.1 BACKGROUND

In recent years, extraordinary strides have been made in the development and application of synthetic scene generation technology. For example, real-time computer image generation (CIG) visual systems capable of simulating scenes viewed by a flight crew either out-the-window or via imaging sensor displays (e.g., FLIR, LLLTV, radar, etc.) are being employed extensively and very effectively in many training-related applications. In addition, CIG technology is being utilized in nonreal-time image prediction systems for such applications as mission planning, war gaming, and missile terminal guidance. An example of the mission planning application is the Radar Prediction System (RAPS) which is undergoing full-scale development at General Electric's Simulation and Control Systems Department.

With flight missions being attempted at increasingly lower altitudes and with the requirements for faithfully simulating/predicting the characteristics of advanced sensors becoming ever more exacting, the scene synthesis problem is becoming proportionately more complex. At low altitudes, the displayed/viewed scene content must usually be significantly higher if the flight crew is to perceive the visual cues that are vital for navigation and target detection/identification in an unfamiliar and dynamically changing environment. Given that current "conventional" CIG techniques and hardware implementations are employed, this higher scene content typically translates to attendant higher content in the data bases which model the environment, larger memories for storage of the data bases, and increased processing for generation of visual/sensor scenes. All of this equates to significantly higher costs for hardware, software, and data base preparation.

Considering the visual/sensor image predictions generated in nonreal-time will suffice for many military operational applications, the attendant digital processing requirements are not nearly so severe. Nonetheless, the impact of higher scene complexity on data bases (and storage thereof) is not necessarily lessened. It would seem logical, then, that means be sought to achieve low altitude scene representations which are more "digitally compact". That is, only the information that is essential to the particular tasks or operations being performed would be displayed; nonessential scene detail would be minimized or eliminated. In light of such a consideration, research has been undertaken under Government sponsorship to determine if reduction of

prediction image content to only that needed for achievement of acceptable flight crew performance is feasible and practical. The results to-date (entry P.3 in Topical Bibliography, Reference II) are encouraging; however, more research is required before firm conclusions can be drawn.

1.2 OBJECTIVE

The purpose of the research effort covered in this report was to investigate, identify, and develop technology necessary to support efficient computer image generation (CIG) processes associated with low altitude, sensor-related scenes. While emphasis was originally intended to be on applying image analysis techniques to obtain digitally efficient low altitude scene representations which could be established through automatic scene processing, the thrust was shifted more towards the processing associated with image generation rather than image analysis. This was done with the concurrence of the RADC Technical Officer for the study, and justified on the basis of the following rationale:

a. Results of related studies addressing the human factors aspects of reducing visual/sensor scene content to minimum acceptable levels were not available in time for consideration in this study.

b. Rather than attempting to carry on an effort duplicating that mentioned in Item a., it was estimated that General Electric could contribute more significantly to the image generation processing and data base handling aspects of the study owing to past and current in-depth experience and expertise in these areas.

c. The current or anticipated availability of implementation techniques, architectures, devices, and/or components which have potential for softening the impact of high scene content deserve equal consideration in the study and should not be disregarded.

1.3 APPROACH

The methodology or flow employed in conducting the study is shown in Figure 1.3-1, and consists of the four basic tasks discussed in the following paragraphs:

1.3.1 BACKGROUND INVESTIGATION

This initial study activity involved an extensive literature search, and the acquisition, and review/analysis of relevant documents spanning topics of which the following are typical:

- Perceptual psychology in visual training.
- Artificial vision and image processing.
- FLIR, LLLTV, microwave, and other visual sensor displays.

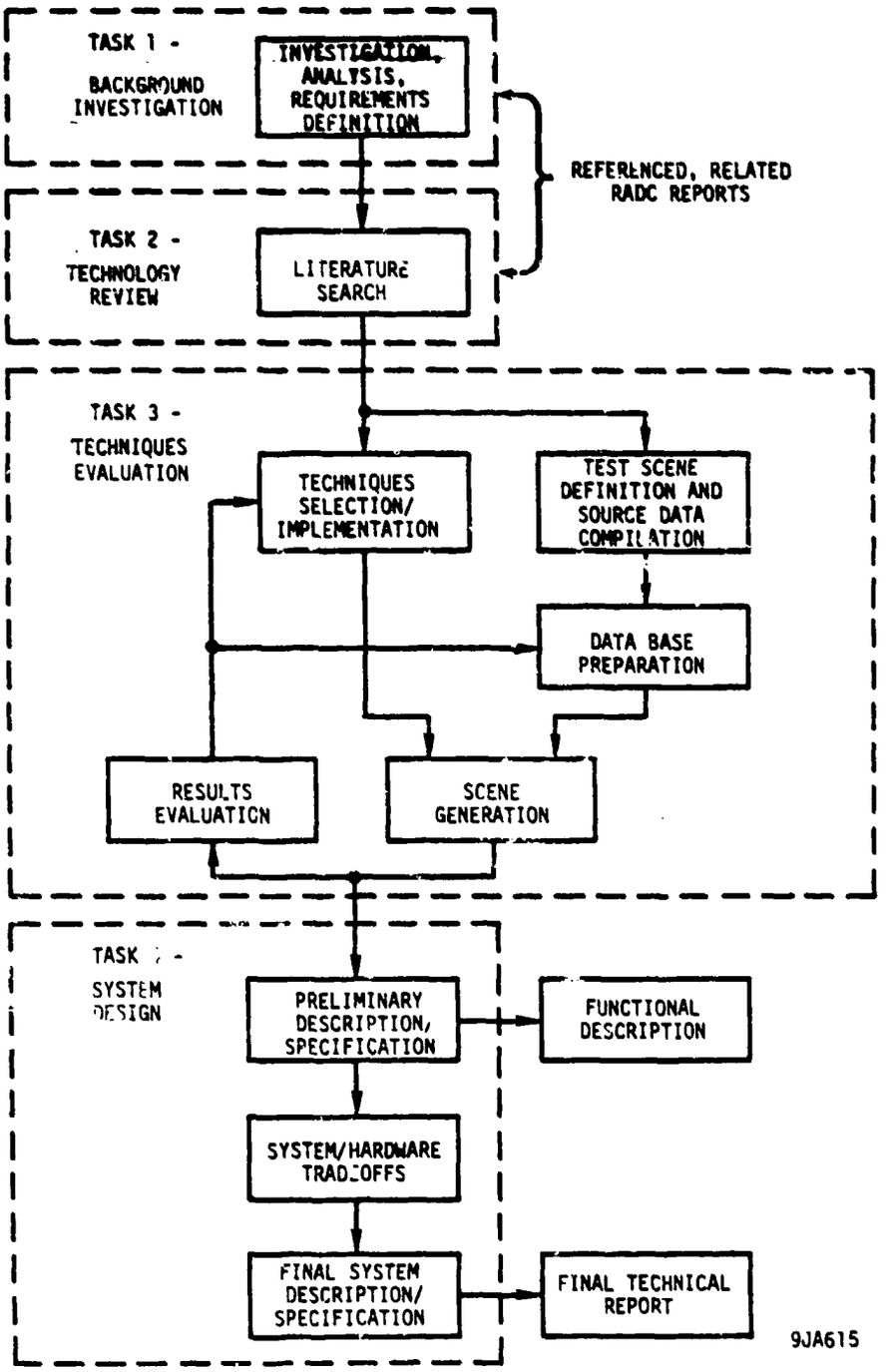


Figure 1.3-1. AWS Sensor Prediction Techniques Study Flow

- Computer graphics.
- Computer image generation and simulation.
- Grid data base display and generation.

Specific references within each of these categories are listed in the Topical Bibliography at the end of this report. In addition, many of these references were used to formulate the Review of Key References (annotated bibliography) also provided herein.

An effort conducted concurrently with the literature search and analysis activity was that of identifying generalized requirements for visual/sensor scene content. The source material proving to be of the most value was Government specifications contained within requests for proposals, and analyses performed by General Electric in response. The significant results of both the background investigation/analysis and requirements definition are presented in Section 2 of this report.

1.3.2 TECHNOLOGY REVIEW

Performance of the literature analysis and requirements definition activities in the previous task provided a logical lead-in to the effort to identify existing technologies and techniques worthy of consideration for experimental and/or analytical evaluation in the next task. More specifically, the principal aim was to determine the most promising means for realizing efficiencies in scene generation. In addition, however, a secondary goal was to compile for presentation the various considerations or factors which must be taken into account in synthesizing scenes for display using a digital data base. The significant results of this exercise are also outlined in Section 2 of this report.

1.3.3 TECHNIQUES EVALUATION

As shown in Figure 1.3-1, this task consists of several steps or subtasks, namely, techniques selection, test scene definition, data base acquisition, scene generation, and results evaluation.

1.3.3.1 Techniques Selection

On the basis of the results of previous task results, final selection of candidate techniques and/or algorithms for implementation and experimental or analytical evaluation was made. The key criterion for selection of a given candidate was its estimated potential for minimizing storage and processing loads while preserving the scene content necessary for effective simulation/prediction. Following selection of techniques for evaluation, the required software was accumulated.

1.3.3.2 Test Scene Definition

This subtask involved the selection and definition of test scenes which would enable the subjective evaluation of both the images produced and the algorithms/techniques employed to generate them. Key to the selection/definition process were considerations as to attributes which the test scenes were required to contain.

1.3.3.3 Data Base Acquisition

Having selected the techniques to be evaluated and defined the test scenes to be used in the evaluation process, the necessary data bases were acquired. Owing to General Electric's extensive simulation-related development and contractual activities, data bases appropriate to this study effort were readily available with little or no modification required.

1.3.3.4 Scene Generation

Following completion of the technique selection and data base acquisition, actual test scenes were generated using both the Static Image Generation facility currently in operation at General Electric's Advanced Technologies Laboratory, and a real-time CIG visual system being developed for the U.S. Air Force. A laboratory system developed to evaluate image generation algorithms and data base editing techniques, the Static Image Generation facility fulfilled a key role in implementing the experimental phase of this study. A description of the Static Image Generation system is provided in Section 3 of this document.

This subtask, together with the preceding techniques selection and scene generation subtasks, constituted an iterative process aimed toward optimization of techniques/algorithms for achievement of the defined project objectives. This evaluation involved qualitative judgments as to scene quality in terms of estimated effectiveness for low-level simulation/prediction, and quantitative judgments relative to effects on processing and data base efficiency. This evaluation process amounted to the trading off of image effectiveness versus processing/storage required and resulted in the identification of promising techniques requiring further investigation/experimentation.

1.3.4 SYSTEM DESIGN

The final phase of the study involved the preliminary design of an advanced image synthesis research system to ultimately be employed in the further research and development of low-altitude data base technology and corresponding scene generation techniques. This activity consisted of the following subtasks:

a. System requirements definition.
b. Candidate architectures investigation.
c. Candidate display systems survey.
d. Candidate peripheral devices survey.
e. System description/specification.

The performance of these subtasks resulted in the publication and submittal, as a contract deliverable item, of the "Functional Description for an Advanced Image Synthesis System" document. In general, the content of this document is summarized as follows:

a. System Performance Requirements - A statement of the required performance parameters (e.g., time allowed for generation of a scene, growth provisions, color display resolution, etc.) for the system.

b. Trade-off Study Results - A discussion of the various system architectures and components (hardware and software) that were considered, including rationale for selection of one approach/component over another provided.

c. System Overview - A top level description of the system, including block diagrams and functional flows enabling an overall conceptual understanding.

d. Hardware Description - Coverage of functional capabilities and block diagrams of hardware components with facilities data provided where appropriate.

e. Software Description - A description of system software (i.e., operating systems), support software (e.g., compilers, assemblers, etc.), and applications software.

f. Performance - A discussion of the estimated performance of the system, with emphasis on scene generation time, storage capacity, and system flexibility (ease of programming, growth provisions, etc.).

1.4 REPORT ORGANIZATION AND SCOPE

Section 2 of this report covers the results of performing Tasks 1 and 2 which are the Background Investigation and Technology Review efforts, respectively. The principal product of Task 1 is a "bottom line" estimate of the necessary input to a visual/sensor scene generation system upon which all processing and storage requirements are predicated. Next, a survey of scene generation, display, and analysis technology/techniques is presented.

A critical evaluation of scene synthesis techniques is covered in Section 3. This evaluation is based on analysis and on experimentation carried out in the General Electric Static Scene Generation laboratory facility. Color photographs of video images are included in the report.

In Section 4, a functional description of laboratory research system intended for AWS sensor simulation/prediction techniques evaluation is provided, with hardware, software, and functional performance described in detail.

Finally, conclusions and recommendations resulting from this study are presented in Section 5. A comprehensive topically organized bibliography compiled during the study effort is supplied as item II in the Reference Material. References to entries within it are given in parenthesis indicating topic and entry; for example, V.3 refers to entry number 3 under topic V - Artificial Vision and Image Processing. A review of selected references is provided in item III in the reference material.

Volume 2 of this document contains reprints of three relevant reports which were judged to be of sufficient value for inclusion.

SECTION 2

BACKGROUND INVESTIGATION AND TECHNOLOGY REVIEW

2.1 BACKGROUND INVESTIGATION

2.1.1 VISUAL/SENSOR SCENE REPRESENTATION

The logical first step in an effort to investigate techniques for synthesizing visual/sensor scenes in a digitally efficient manner is that of defining baseline requirements for the attributes which those scenes must exhibit. Ideally, it should be possible to characterize those attributes for a range of simulation or prediction system mission applications in both quantitative and qualitative terms. Having done this, candidate image processing and/or scene generation techniques can then be evaluated in light of their ability to reproduce the desired scene characteristics while minimizing the costs and other factors associated with processing, storage, and data base preparation.

The key factor in defining the necessary scene attributes is that of visual perception. However, an extensive literature search conducted in the course of this study revealed that while a great deal of information is available concerning the fundamental sensitivities of the human visual system, little is known quantitatively about the perceptual process of acquiring information from a scene.

Although there seems to be a general view among current and potential users of simulator and/or prediction system devices that effectiveness is directly influenced by fidelity of the synthesized scenes, there is evidence (P. 3, F. 15) that this view is questionable. In practice, the "stylizing" of visual scenes involving departures from the mathematical reality of the world has in many cases proved to improve rather detract from the performance of simulators as measured by training transfer. It is clear, however, that the "realism versus effectiveness" issue will continue to be controversial and largely a matter of subjective judgement. In any event, acceptable systematic methodologies for specifying scene content parameters or variables do not appear to exist at the present time. However, there are certain minimal fundamental scene attributes which are obviously required in any simulated visual/sensor scene. Briefly stated, these attributes are:

1. A scene should have texture so that surfaces will have a solid appearance.
2. A scene should contain familiar objects, represented in either pictorial or symbolic form.
3. A scene should obey the laws of perspective transformation.

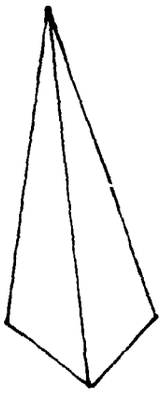
Texture provides important visual cues as to the tilt of the ground plane and the relationships among surfaces and objects, since natural surfaces nearly always possess perceivable texture. Although texture may not be necessary in every mission application, a visual/sensor prediction or scene generation system should have this capability available when needed.

Another important source of information is the presence and arrangement of familiar objects. These establish locations on the ground, and provide important visual cues for distance and size. Trees, car, aircraft, roads, railroad tracks, and houses have general size distributions. The amount of detail to which these objects must be portrayed is an important consideration; it is probably only necessary to provide sufficient detail so that an object can be recognized and distinguished from similar objects. It may sometimes be desirable to store the same o ect at a number of levels of detail as illustrated in Figure 2.1-1. The displayed level of detail can be either an input parameter or a function of the distance between the viewpoint and the object's centroid. These and other considerations such as whether all similar objects (e.g., trees) be given identical size and shape, and whether objects should be arranged in random or regular patterns are perhaps best left to the discretion of the scene modeller.

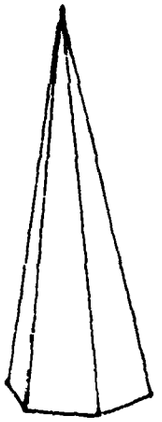
The perspective transformation requirement is fundamental. Although perhaps not as critical for prediction or "non-dynamic" applications, humans have a natural inclination for seeing in depth (three-dimensional space). The synthesis of two-dimensional information does not allow the third dimension to be perceived; consequently, sensors which produce perspective images of three-dimensional images scenes (visual, FLIR, LLLTV) must in general be simulated in three-dimensional format. A possible exception may be the representation of certain objects (trees, vehicles, boulders, buildings, etc.) by means of two-dimensional symbols placed in correct 3-D position (just as, for example, a billboard in the real world is a flat representation of a 3-D scene) in cases where it is only necessary to know what the object is and where it is located.

In a very real sense, scene content requirements are heavily dependent on the prediction system application, and somewhat less dependent on the type of sensor being simulated in the scene. For most high altitude applications, the characteristics of manmade and vegetation features are of little importance in terms of providing cues, thus scene detail need not be high. On the other hand, low-level nap-of-earth or on-ground type missions generally require that terrain features be accurately portrayed if maximum advantage is to be taken of cover and concealment and if navigation and target detection and identification is to be successful. For example, navigation in low-level flight usually involves a visual "pattern matching" process in which map features are compared with terrain and culture features being viewed via sensors or out-the-window.

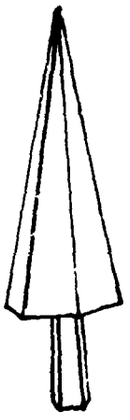
Scene content requirements for IR and LLLTV simulations are similar in many respects even though the characteristics of both sensors change with time-of-day, atmospheric conditions, and seasons. The scene content of a simulated sensor image must as a minimum include the kinds of features that normally stand out in a real image of its respective type. For example, the warmer areas of moving vehicles are conspicuous in FLIR imagery. An LLLTV scene must account for the absence of sun and the addition of active light sources. DeGroot (F.6) infers that patterns of light are particularly important in FLIR displays, while the shape of objects is important in LLLTV.



6 Edges
4 Faces
4 Vertices



12 Edges
7 Faces
7 Vertices



32 Edges
16 Faces
20 Vertices

Figure 2.1-1. An Evergreen Tree Represented at Three Levels of Detail

For some military operations, it may be desirable to know how the same scene would appear on different sensor systems. In that event it is essential that the simulated imagery faithfully replicate the real imagery that would appear under actual conditions. Correlation of radar targets and their visually displayed counterparts can be accomplished through the use of separate yet similar data bases. (The special processing required to simulate radar scenes is beyond the scope of this study; rather, the reader is referred to references F.2 and F.3 .) Data bases very similar to those created for visual scenes may be used for IR and LLLTV; this will help ensure that the required correlations can be effected.

The need for slightly different data base content for the different classes of imagery is illustrated by example: a visual data base modeller may construct a truck from a set of faces, all with the same color; a FLIR data base modeller may note the dichotomy in temperature between the front and rear of the truck and treat each separately. The regions above and below the oil line in an oil tank must also be treated independently in a FLIR data base. Each surface of an object in a visual data base may be given a different color, however this is not likely to be necessary in an LLLTV data base.

In all cases it will be the system software, firmware, and/or hardware which performs the necessary modifications to the generated imagery to model the distortions inherent with each sensor type. Such transformations and distortions need not be incorporated into the data base. That is, though displays of the same environment may differ greatly on FLIR and LLLTV, the quantity and structure of the data is similar in both cases. The overall geometric structure of an object is the same for the simulation of both types of sensors. Qualitative attributes associated with the object correspond to color in the visual domain and temperature in the IR domain. Although display simulation may process such attributes rather differently, their digital representation is nearly identical. It is standard practice in CIG systems to use environment data from the same source for both visual and sensor display simulation. Level of detail may differ slightly to accommodate sensor resolution and field of view. Qualitative attributes (color, temperature) are often processed by simply applying a display color table appropriate to the sensor and time of day; training transfer appears to be quite sufficient using this approach.

2.1.2 DATA BASE CONTENT/DETAIL

Maps and charts are extremely critical to low-level flight navigation since the pilot/navigator typically must correlate significant map features with those actually perceived out-the-window or via imaging sensor displays. Accordingly, charts intended for navigational use are not simply renderings of geographical areas; rather, they are summaries of specific terrain and ground features which are readily identifiable and significant for navigational purposes. That is, human factors aspects are specifically taken into account in the cartographic process. For this reason, such charts are valua valuable in defining requirements for the content of data bases from which visual/sensor scenes are to be generated. A general rule, then, is that "whatever is contained in the map/chart should also be contained in the data base".

To illustrate the processes involved in defining scene content and attendant data base storage requirements for simulation of a typical tactical operating environment, an example analysis is presented. This example is modelled after an exercise conducted by General Electric to define the data base requirements for an aircraft weapon system simulator system, and involved both high and low altitude flight considerations. However, only the low altitude aspect is considered here.

In the analysis to follow, the basic unit of measure is "edge count", the contributions to which arise from several sources such as vehicles, airfields, and large geographic regions. The largest overall area of consideration is assumed to be 10,000 square nautical miles (or nm^2). This area, which need not necessarily be square or rectangular, is defined as a "Low Altitude Tactical Region". It is assumed that nested within this region there will be subregions, which for this exercise are defined as "Weapon Delivery" regions and "Target Area" regions. These three types of regions are further defined as follows:

Region Type	Operating Altitudes*(feet)	Operating Area (nm^2)
Low Altitude Tactical	200 - 1000	10,000
Weapon Delivery	100	1,500
Target Area	100	19

*(Height above ground level)

The first task in the exercise involves estimating the number of edges per unit area or unit length required to represent the features expected to be encompassed within a region type and then multiplying this by the area encompassed by each. Once this is done, the total edge count can be used to calculate the data base storage requirement. Edge densities are estimated by sampling the features in representative $25nm^2$ regions of 1:50,000 scale maps (e.g., Defense Mapping Agency Topographic Charts, etc.) for the mission operations area.

The results of the map analysis are presented in Table 2.1.2-1 which is composed of subtables (a), (b), and (c) covering linear features, discrete features and terrain, and target feature categories, respectively. The data in the subtables will now be used to calculate the total edge count for each region type.

Low Altitude Tactical Region

This region is constituted as follows:

- Linear features (441 edges, from table)
- Discrete features and terrain (417 edges, from table)
- Trees
- Small buildings

Based on assuming trees (12 edges) to be interspersed at 1500 foot intervals (16 per square mile) over $10nm^2$ of a $25nm^2$ area, the edge count is:

12 edges x 16 trees/nm^2 x $10nm^2$ area = 1920 edges

Assuming small buildings (15 edges) to be interspersed at 1500 foot intervals (16 per square mile) over $9nm^2$ of a $25nm^2$ area, the edge count is:

15 edges x 16 buildings/nm^2 x $9nm^2$ area = 2160 edges

The total edge count per $25nm^2$ area is obtained as follows:

1920 tree edges
2160 small building edges
441 linear feature edges
417 discrete feature and terrain edges
4938 total edges per $25nm^2$

Finally, the total edge count for the Low Altitude Tactical Region ($10,000nm^2$) is calculated as follows:

Table 2.1.2-1(a) Linear Feature Edges

Feature Type	Edges Per Mile	Total Miles	No. of Edges
4-Lane Highways	1.5	8	12
2 & 3 - Lane Roads	2	20	40
2-Lane Light Dirt Roads	6	45	270
Railroads	1.2	8	10
Power Lines	2	2.5	5
Coastal Boundaries	1.5	16	24
Inlets & Rivers	8	10	80
		Total edges per $25nm^2$:	441

Table 2.1.2-1(b) Discrete Feature and Terrain Edges

Feature Type	Edges Per Mile	Total Miles	No. of Edges
Towns/House Clusters	-	-	30
Airports	14	1	14
Piers & Dams	4	37	148
Bridges	4	4	16
Churches	8	20	160
Land Types	-	-	45
Terrain	-	-	4
Total edges per $25nm^2$:			417

Table 2.1.2-1(c) Target Feature Edges

Feature Type	No. of Edges
Trails	4
SAM Sites	30
Radars	30
Vehicles	50
Buildings	45
Total edges per area (rectilinear)	159

Note: Total edges per circular area $159/\pi$ 50.6

$$4938 \text{ edges} \times 1/25\text{nm}^2 = 198 \text{ edges/nm}^2 \text{ density}$$

$$198 \text{ edges/nm}^2 \times 10{,}000\text{nm}^2 = \underline{1{,}980{,}000 \text{ edges}}$$

Weapon Delivery Region

This region is composed of the feature content for the Low Altitude Tactical Navigation Region, above, plus additional unimproved dirt roads. In this example, around 20 miles of crooked roads provided a contribution of 400 new edges per 25nm^2 (or 16 edges per nm^2). This edge density is added to the 198 edges per nm^2 figure as follows:

$$16 + 198 = 214 \text{ edges/nm}^2$$

The edge count for the Weapon Delivery Region is thus:

$$214 \text{ edges/nm}^2 \times 1500\text{nm}^2 = \underline{321{,}000 \text{ edges}}$$

Target Area Region

The data base content for this region includes all higher level data previously estimated, plus the features listed in Table 2.1.2-1(c) and additional trees inserted every 200 feet to avoid the undesirable training effects of targets appearing in sparse areas.

Adding the edge density for the circular Target Areas yields:

$$51 \text{ edges/nm}^2 + 214 \text{ edges/nm}^2 = 265 \text{ edges/nm}^2$$

At 12 edges per tree, this contribution yields 11,090 edges/nm^2. Thus, the data base content for the 19nm^2 Target Area is calculated:

$$19\text{nm}^2 (11{,}090 \text{ edges/nm}^2 + 265 \text{ edges/nm}^2) = \underline{215{,}745 \text{ edges}}$$

The edge density figures discussed above were derived from considerations of minimum content requirements for low level flight. In some situations there may be additional visual cues required for adequate perception of orientation and depth which can be provided by texture as described in paragraph 3.1.2.4. To appreciate the need for texture, consider the visual density of a scene viewed directly from above at an altitude of 1000 feet with a sixty degree field of view. The ground area enclosed by the field of view is a square whose sides are 1154 feet long. This encloses an area of .036nm^2. Hence, for target area densities of 265 edges/nm^2, .036 x 265 = 9.5 edges. That is, on the average such a scene will barely include the number of edges to display a cube. Texture patterns can enrich the scene and provide visual cues equivalent to fifty to over athousand edges in such situations.

Data Base Storage Requirements

Data base content for the total Low Altitude environment is computed as follows:

Low Altitude Tactical Region	1,980,000 edges
Weapon Delivery Region	321,000 edges
Target Area Region	215,745 edges
Total	2,516,745 edges

The total edge content is doubled to account for level-of-detail:

2,516,745 x 2 = 5,033,490 edges

Finally, the grand total edge count figure is multiplied by the number of bytes required to store each edge (which is 48 in a typical CIG-based simulator). Thus:

$$5,033,490 \times 48 = \underline{241.6 \text{ megabytes}}$$

2.1.3 SCENE/DATA BASE VIEWING

2.1.3.1 Data Base Area Versus Viewing Range

To quantify scene complexity, the relationship between the pilot's altitude and the potentially viewable area of the spherical surface of the Earth must be known. Then, the number of edges that are required to depict the terrain, culture, and 3-D objects within the viewing area must be determined. The number of edges required to depict a scene, however, is of course not some absolute specifiable value; this depends upon both the level of detail to which the scene is to be modelled and the intricacy of the scene itself.

(A) Spherical Viewing Area Computation

It can be shown that the area of the spherical viewing surface varies linearly with height above the Earth, if this height is small compared to the Earth's radius. First, several terms are defined as follows (see Figure 2.1.3.1-1):

H = altitude above the Earth of the observer's eyepoint
R = radius of the Earth
N = nadir
T = point of Earth tangency to line-of-sight
S = length of arc connecting N and T (in a slight misuse of notation the term S will be used to denote the arc itself).
$\emptyset$ = angle in radians at geocenter between nadir and T
r = distance in Y-direction between Earth's center and T

The following relationship can then be written:

$$\emptyset = \cos^{-1}\left[R/(R+H)\right] \quad \text{and} \quad S = R\emptyset.$$

If the arc S is rotated about the Y-axis, a surface is generated which corresponds to the surface area which is visible at altitude H. The area of this surface is denoted by A. The value of A is derived below.

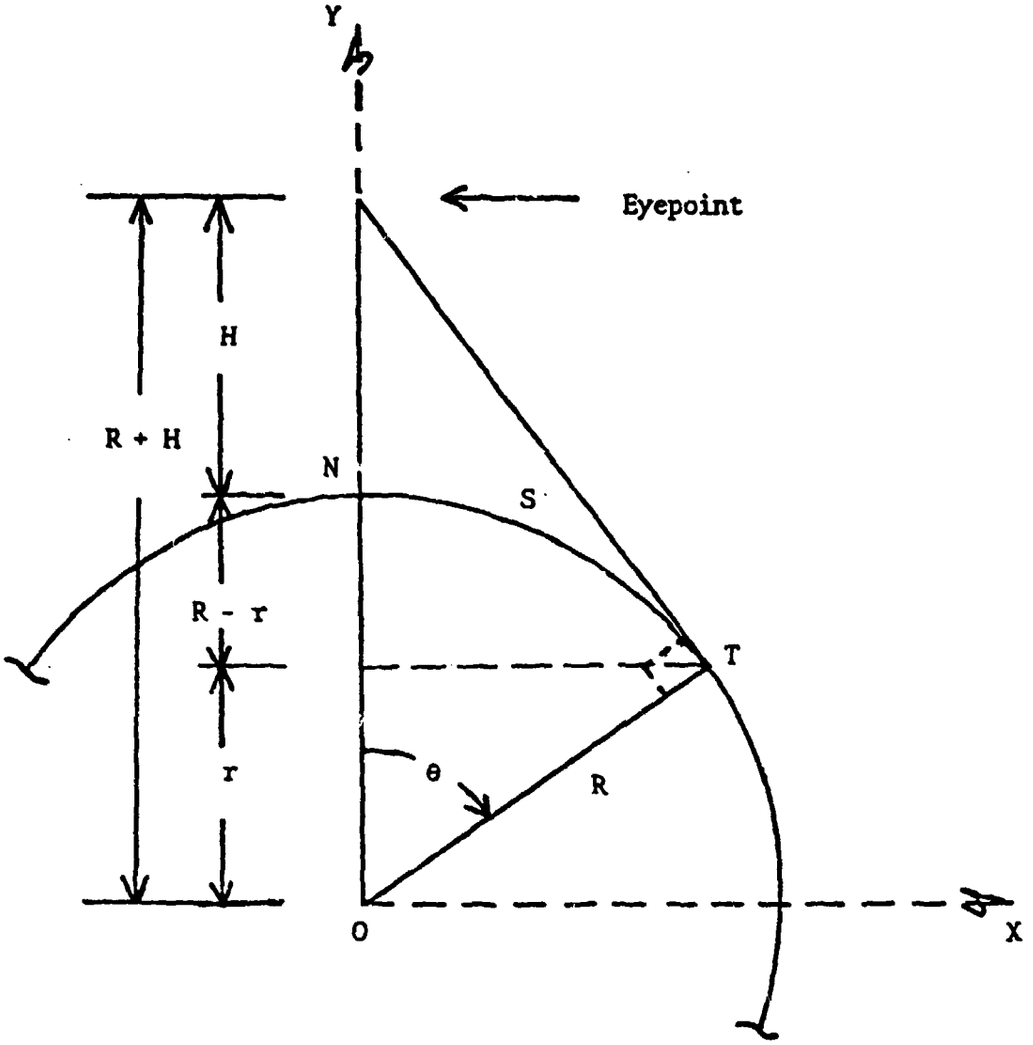


Figure 2.1.3.1-1. Visibility of Earth's Surface as a Function of Observer Altitude

$$A = 2\pi \int_{Y=r}^{Y=R} X \sqrt{1+(dX/dY)^2}\, dY \qquad \text{Where } X = \sqrt{R^2 - Y^2}$$

$$A = 2\pi \int_{Y=r}^{Y=R} \sqrt{R^2 - Y^2} \sqrt{1+\left[-Y \sqrt{R^2 - Y^2}\right]}\, dy = 2\pi R\,(R-r)$$

Substituting $r = R\cos\phi$, results in:

$$A = 2\pi R^2 \left[1-\cos\left(\cos^{-1}\left[R/(R+H)\right]\right)\right] = 2\pi R^2 H/(R+H)$$

For low altitude, $H \ll R$ and $A \simeq 2\ RH$.

The equations derived above can be used to obtain Table 2.1.3.1-1.

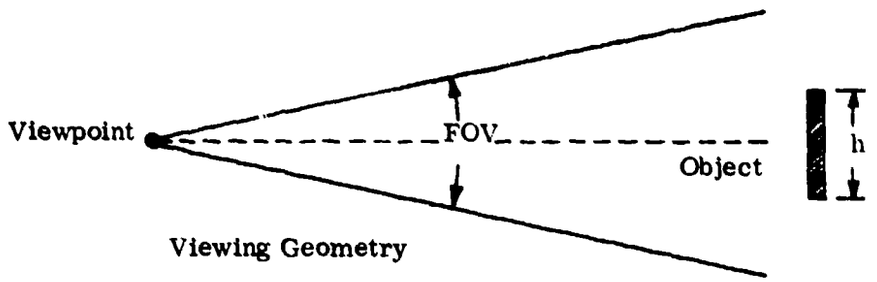
Altitude, H (Feet)	Arc Length, S (Miles)	Surface Area, A (Sq. Miles)
1	1.22	4.71
2	1.73	9.43
3	2.12	14.15
4	2.45	18.87
5	2.74	23.58
6	3.00	28.30
7	3.24	33.01
8	3.47	37.73
9	3.68	42.45
10	3.87	47.16
50	8.66	235.82
100	12.25	471.63
200	17.33	943.26
500	27.40	2358.13
1000	38.75	4716.14
5000	86.63	23576.19
10000	122.50	47141.11
100000	386.69	469393.32

Table 2.1.3.1-1. Visible Area of Earth as a Function of Altitude

The areas in the table represent the maximum viewable surface under ideal viewing conditions. Atmospheric attenuation and the resolution of the display device decrease visual content as described below.

(B) Scan Line Resolution Effects

The number of scan lines constituting an object's image on a display can be determined from the geometry shown below.



Viewing Geometry

The number of raster lines L across an object of projected height h is given by Erickson (P.7) as

$$L = \frac{2N \tan^{-1}(h/2R)}{FOV}$$

where
- h = projected object height
- N = total number of scan lines actually displayed by CRT
- R = range of object
- FOV = field of view of simulated sensor

For small objects, the above equation can be approximated by

$$L = \frac{57.29hN}{R*FOV}$$, where FOV is in degrees and R and N are in meters.

An object's size on a CRT changes with viewing angle. The simulated sensor's azimuth viewing angle, measured with respect to the object's longitudinal axis together with the sensor's elevation viewing angle (looking down), determine projected size. The evaluation for a CRT with horizontal raster lines requires a knowledge of the projected object height h (see Figure 2.2.1-2). This is given by

$$h = h\cos\delta + (L\cos\psi + W\sin\psi)\sin\delta .$$

Erickson and Main (P.10) conducted experiments involving viewing of real scenes of real targets in a cluttered background. The conclusion was that real targets could be detected in the scenes every time with about six raster lines across them. For identification the data were more variable; over 20 raster lines were generally required across a target before it could be identified with certainty. Target size, lighting, aspect angle, and scene scale varied among the scenes displayed.

Scott, Hollanda, and Hartabedian (F.14) conducted a similar experiment with vehicular targets in which the number of scan lines per vehicle and the angle of view were varied on a noiseless display to determine identification and classification accuracy. It was concluded that about 20 scan lines per vehicle is a reasonable number for both a satisfactory level and spread of performance.

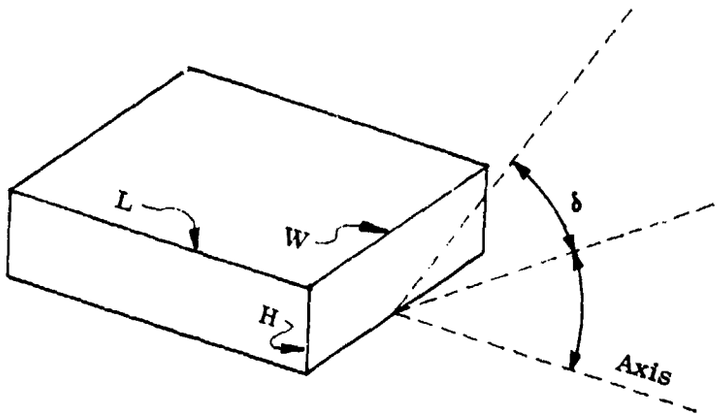
Erickson and Hemingway (P.8, P.9, P. 11) conducted several experiments related to the identification of military vehicles. The number of scan lines per image, type of background, and angular subtense were all varied. A high probability of identification was obtained when there were at least 10 scan lines per vehicle and an angular subtense of over 14 minutes of arc. Performance, however, varied with the type of background.

Erickson (P.7) found that accurate detection of small isolated targets required only three scan lines at 18 percent contrast, but 5 scan lines at 7 percent contrast. Recognition generally required from 9 to 12 scan lines. The same experiment was conducted as a function of the size of the object in terms of minutes of arc. It was found that the detection of an isolated target required 5 arc minutes at 18 percent contrast and 9 arc minutes at 7 percent contrast. Recognition generally required from 10 to 15 arc minutes.

For purposes of this study, it would seem desirable to apply all this data to obtain the range out to which an object must be displayed. However, there are a number of parameters which when varied will produce different results. These are object size, field of view, and number of scan lines subtended by the object. This number of scan lines is a function of whether it is desirable to display an object when it is just large enough to be detected, or whether it should be displayed only when it can be recognized; thus, the choice is task-specific. The range for two rather extreme cases is derived to bound the results:

Case 1: FLIR and LLLTV displays generally have a field of view of between 7 and 20 degrees. To detect an object may require as little as 3 raster lines. An object height of 20 meters is considered:

$$R = \frac{57\ (h)\ (N)}{L\ (FOV)} = \frac{(57)\ (20)\ (512)}{(3)\ (7)} = 28 \text{ kilometers} = 17 \text{ miles}$$



H = actual height
W = actual width
L = actual length
ψ = azimuth viewing angle
δ = elevation viewing angle

Figure 2.1.3.1-2. Projected Object Height

Case 2: A visual scene generally has a field of view of between 30 and 60 degrees. To recognize an object may require 20 raster lines, thus:

$$R = \frac{(57)\ (20)\ (512)}{(20)\ (60)} = 0.48 \text{ kilometers} = 0.3 \text{ miles}$$

Case 3: For a very large object such as a mountain, raster resolution is not a limiting factor.

(C) Atmospheric Attenuation Effects

The visual range of an object is defined as the horizontal distance for which the apparent contrast betwen the object and its surroundings becomes equal to the threshold contrast of the human eye. The slant visibility of an object is the slant distance for which the contrast between the object (which is below the observer) and its surroundings, as seen from an aircraft, is equal to the threshold contrast of the human eye. The background for visual range is the horizon, while the background for slant visibility is terrain and culture.

Case 1: Horizontal Visibility of a Black Object Against the Horizon

The contrast of a black object of apparent luminance B_b against a horizon background of luminance B_h is

$$C = \frac{B_h - B_b}{B_h} e^{-ax}.$$

If the contrast is equal to the threshold value of the human eye (due to distance hazing), the distance x is equal to the visual range V. The threshold value C_{min} equals 0.02 for ordinary subjects in daytime. Substituting this value into the above equation gives the normal visual range of

$$V_n = 3.912/a,$$

Where $V_n = V = x$ for $C = .02$. Typical values of "a" are given in Table 2.1.3.1-2. Substituting the values from Table 2.1.3.1-2 into the above equation tells us that at zero altitude, when there is good visibility, the visual range is approximately from 12 to 39 miles.

The threshold value of C varies with background luminance, increasing during twilight and night. It also depends upon the angular diameter of the object, which must be greater than one-half minute of arc for the above formula to apply.

Height (in feet)	Good Visibility "a" (in meters^{-1})	Poor Visibility "a" (in meters^{-1})
0	0.6 to 2.0 x 10^{-4}	3 to 15 x 10^{-4}
1,000	0.5 to 0.9 x 10^{-4}	3 to 15 x 10^{-4}
10,000	0.3 to 0.7 x 10^{-4}	0
20,000	0.2 to 0.5 x 10^{-4}	0

Table 2.1.3.1-2. Typical Values of Extinction Coefficient "a" as a Function of Altitude.

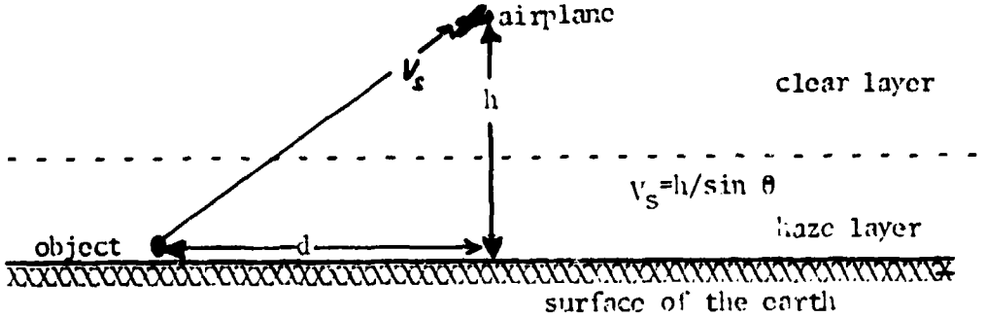
Object Type	Clear Sky	Overcast Sky
Fresh Snow	0.2	1
Desert	1.4	7
Forest	5	25

Table 2.1.3.1-3. Sky-Ground Ratio for Typical Objects.

Case 2: Slant Visibility in a Uniformly Illuminated Atmosphere

Slant visibility is concerned with seeing details in terrain from an aircraft. For this case the luminance of the horizon cannot be used as the background, since it is assumed that the angular distance of the objects of interest from the horizon is large. The main factor in slant visibility is the contrast between an object and its background. Tables can be used to find the albedos of standard objects and typical backgrounds (C.17, C.18).

The geometry to be used is shown below:



The equation to be used for map distance d is

$$d = a^{-1} \ln \left[S^{-1} \left(\frac{A_B - A_O}{A_B C} - 1 \right) + 1 \right]$$

Where A_O = albedo of object

A_B = albedo of background

a = total scattering coefficient for line-of-sight

S = sky-ground ratio (typical values are given in Table 2.1.3.1-3).

For real objects against terrain, a high difference in albedo would not be expected. If $A_O = .9A_B$, then

$$\frac{A_B - A_O}{A_B C} = 0.1/C = 5 \text{ since } C = 0.02 \text{ for the standard observer.}$$

Choosing a clear atmosphere with $1/a = 10^4$ meters and a value of S = 2, then

$$d = 10^4 \ln \left((5-1)/2 + 1 \right) \text{ meters} = 11 \text{ kilometers} = 7 \text{ miles.}$$

A second, rather "worst case" example is considered. Assuming an extremely clear atmosphere with $1/a = 1.6 \times 10^4$ meters, higher contrast objects ($A_0 = .8A_B$), and a value of S = 1.5, then

$$d = 1.6 \times 10^4 \ln \left((10-1)/1.5 + 1 \right) \text{ meters} = 31 \text{ kilometers} = 19 \text{ miles.}$$

(D) Range Limitation Conclusions

The limiting factor when flying extremely low over flat ground and viewing small objects on the ground is the curvature of the Earth. If the scene generation system is limited to scenes of this type, the size of the modelled environment would be quite small. However, such a restriction would reduce the flexibility of the scene generator. For other types of scenes, relatively small objects (e.g., tanks, houses) and very large objects (rough terrain) should be treated separately.

For large mountains silhoutted against the sky, the range limitation is around 39 miles and is due to atmospheric attenuation. For small objects on the ground the range limitation due to atmospheric attenuation is typically around 7 miles. The range limitation due to raster resolution for recognizable objects on a system with a reasonably wide FOV is not more than a few miles.

2.1.3.2 Storage and Processing Requirements

The impact of scene complexity on storage and processing is illustrated by an example in which system requirements will be computed ignoring all known constraints and assuming that the scene will contain all that an observer is likely to want to see. All data that may be required by the image generation hardware will be stored. Objects, terrain, and culture will be displayed at different levels of detail which will correspond to the annular regions shown in Figure 2.1.3.2-1. The proposed scene content for each of these annular regions is given in Table 2.1.3.2-1.

If the data in Table 2.1.3.2-1 is totaled, the following is obtained

Entity	Total Number of Entity	Total Number of 32-Bit Words
Vertices	9.7×10^6	5.4×10^7
Edges	11.7×10^6	11.7×10^6
Face Numbers	7.2×10^6	7.2×10^6
Face Normals	7.2×10^6	2.2×10^7
Face Pointers	7.2×10^6	7.2×10^6
Centroids	1.7×10^6	5.1×10^6
Color Pointers	1.3×10^6	1.3×10^6
Texture Pointers	6.3×10^5	6.3×10^5
Texture Direction Vectors	1.8×10^5	5.4×10^5
Total		1.1×10^8 32-bit words, or 4.4×10^8 bytes.

Assuming a 20 percent storage overhead, this amount of data should fit onto two 300 megabyte disks. However, this data represents the requirements for only a single small region. If for example a series of snapshots along a route are to be produced, around 100 times this amount of data must be stored. That is, two hundred 300 megabyte disks would be required.

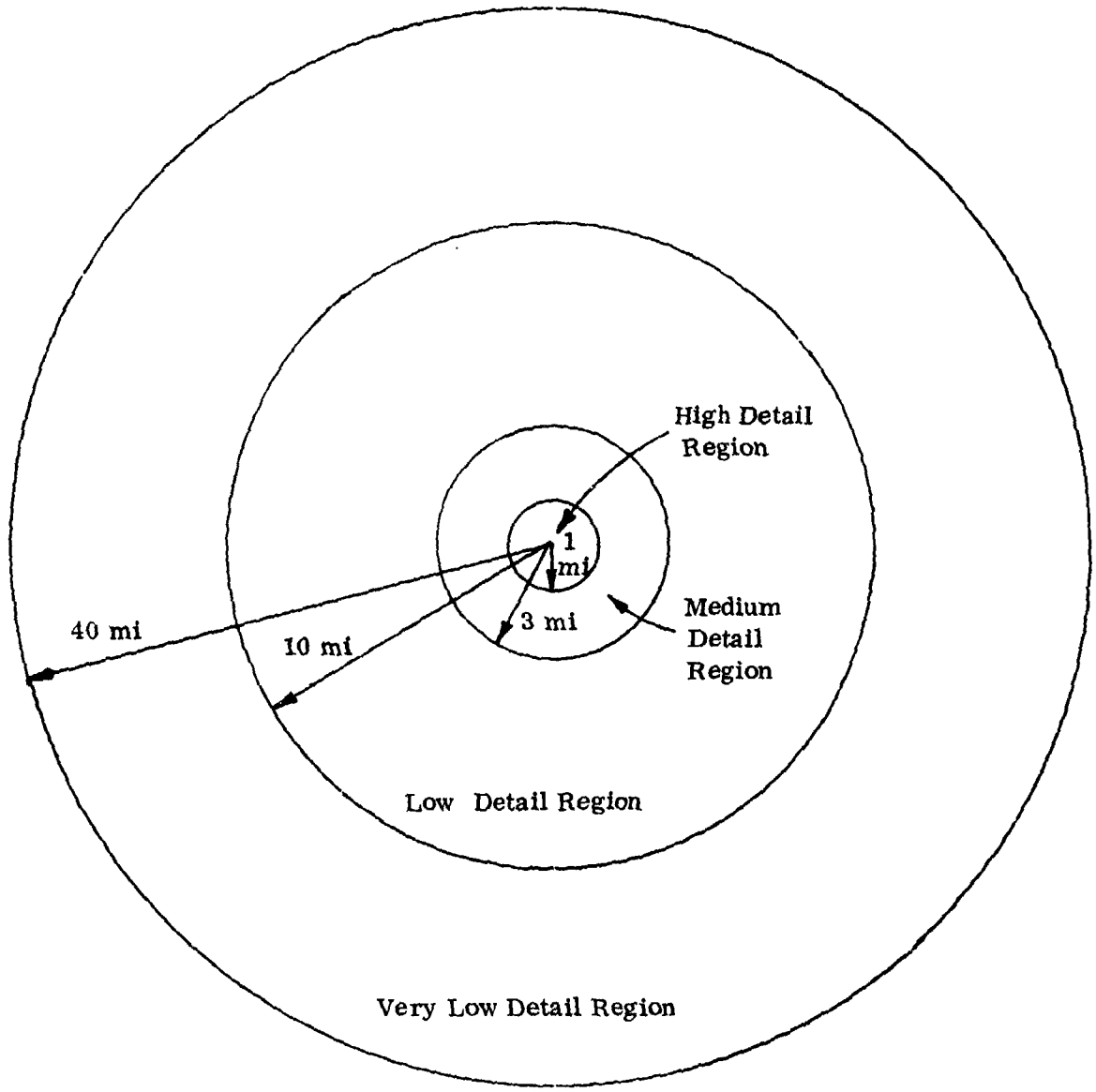


Figure 2.1.3.2-1. Annular Level-of-Detail Regions

Table 2.1.3.2-1. Scene Contents Per Acre

Range (miles)	Acres Within Annular Region	Trees	Buildings	Other Objects	Terrain	Culture
0-1	2×10^3	3 High Detail Trees	1 High Detail Building	1 High Detail Object	1 Terrain Triangle	1 Detailed Culture Polygon
	Vertices	60	20	20	3	7
	Edges	96	30	30	3	7
	Face Numbers	48	15	15	1	-
	Face Normals	48	15	15	1	-
	Face Pointers	48	15	15	1	-
	Centroids	3	1	1	1	-
	Color Pointers	2	2	2	1	1
	Texture Pointers	0	0	0	1	1
	Texture Direction Vectors	0	0	0	3	3

Table 2.1.3.2-1. *Scene Contents Per Acre (Cont.)*

Range (miles)	Acres Within Annular Region	Trees	Building	Other Objects	Terrain	Culture
1-3	18×10^3	3 Medium Detail Trees	1 Medium Detail Building	1 Medium Detail Object	1/2 Terrain Triangle	1 Medium Detail Culture Polygon
	Vertices	21	10	10	3/2	3
	Edges	36	21	21	3/2	3
	Face Numbers	21	7	7	1/2	-
	Face Normals	21	7	7	1/2	-
	Face Pointer	21	7	7	1/2	-
	Centroids	3	1	1	1/2	-
	Color Pointer	1	1	1	1/2	1
	Texture Pointer	0	0	0	1/2	1
	Texture Direction Vectors	0	0	0	3/2	3

Table 2.1.3.2-1. Scene Contents Per Acre (Cont.)

Range (miles)	Acres Within Annular Region	Trees	Buildings	Other Objects	Terrain	Culture
3-10	2×10^5	3 Low Detail Trees	3 Low Detail Buildings	3 Low Detail Objects	1/4 Terrain Triangle	1 Low Detail Polygon
	Vertices	12	8	8	3/4	2
	Edges	18	12	12	3/4	1
	Face Numbers	18	6	6	1/4	-
	Face Normals	18	6	6	1/4	-
	Face Pointers	18	6	6	1/4	-
	Centroids	3	1	1	1/4	1
	Color Pointer	3	1	1	1/4	1
	Texture Pointer	0	0	0	1/4	3
	Texture Direction Vectors	0	0	0	3/4	

Table 2.1.3.2-1. Scene Contents Per Acre (Cont.)

Range (miles)	Acres Within Annular Region	Trees	Buildings	Other Objects	Terrain	Culture
10-40	33×10^5	No Trees	No Buildings	No Objects	1/8 Terrain Triangle	No Culture
	Vertices				3/8	
	Edges				3/8	
	Face Numbers				1/8	
	Face Normals				1/8	
	Face Pointer				1/8	
	Centroids				1/8	
	Color Pointer				0 *	
	Texture Pointer				0 **	
	Texture Direction Vectors				0 **	
	* Standard Color Used					
	** Texture of Standard Type and Orientation Used					

To reduce costs, data from any disk could be rolled onto a 3600 ft, 6250 bpi, magnetic tape. Transfer from disk-to-tape or tape-to-disk would take about six minutes, assuming a 781 kb/sec transfer rate.

With a 45 degree field of view, about one-eighth of the data corresponding to a circular viewing area would have to be read into the computer's core memory. That is, 1.4×10^7 32-bit words would have to be kept in core. This value is one to two orders of magnitude higher than what is reasonable.

Again assuming a 45 degree field of view, about one-eighth of the total number of edges would have to be processed to produce a scene. That is, the required processing capability would be about 1.4×10^6 edges. This value is two orders of magnitude higher than what is currently feasible with a medium size general-purpose computer.

The conclusion to be drawn from the above analysis is the there are practical limits to achieving high scene content in simulated scenes. While the real world is infinitely variable and complex, simulations must depict the world in a simplified fashion. The following paragraphs will show how this can done.

2.1.4 SCENE COMPLEXITY REDUCTION (LEVEL OF DETAIL)

2.1.4.1 Environment Partitioning

It has been shown that 3-D objects need be displayed out to only a distance of a few miles, whereas terrain should be displayed out to about 39 miles. However, there is no need to store and display distant terrain at the same level of detail as nearby terrain, inferring that terrain should be displayed at different levels of detail. One approach is to partition the environment into data blocks. A good block size is 15' latitude by 15N' longitude, which is a common map size. The N value must be a function of latitude, but will be unity (1) for most areas of interest (see Table 2.1.4.1-1). A block will be about 15 miles on a side near the equator.

A suggested arrangement of blocks is shown in Figure 2.1.4.1-1, with pilot's viewpoint being anywhere within the center block. The contents of the blocks would be as follows:

LOD N	=	high level of detail	3-D objects plus high detail terrain and culture
LOD N+1	=	medium level of detail	medium detail terrain and culture
LOD N+2	=	low level of detail	low detail terrain

Lower level of detail blocks can also be defined to correspond to higher viewpoint altitudes.

This arrangement will result in an economical utilization of system capacity. Scene components that are not visible to the pilot, or which are too detailed to be perceived, will not be displayed.

Latitude Zone	Block Size Lat x Long (arc minutes)	DMA Grid Spacing (in arc seconds) Level I	Level II
0° - 50°	15 x 15	3 x 3	1 x 1
50° - 70°	15 x 30	3 x 6	1 x 2
70° - 75°	15 x 45	3 x 9	1 x 3
75° - 80°	15 x 60	3 x 12	1 x 4
80° - 90°	15 x 90	3 x 18	1 x 6

Table 2.1.4.1-1. Suggested Block Size as a Function of Latitude Zone

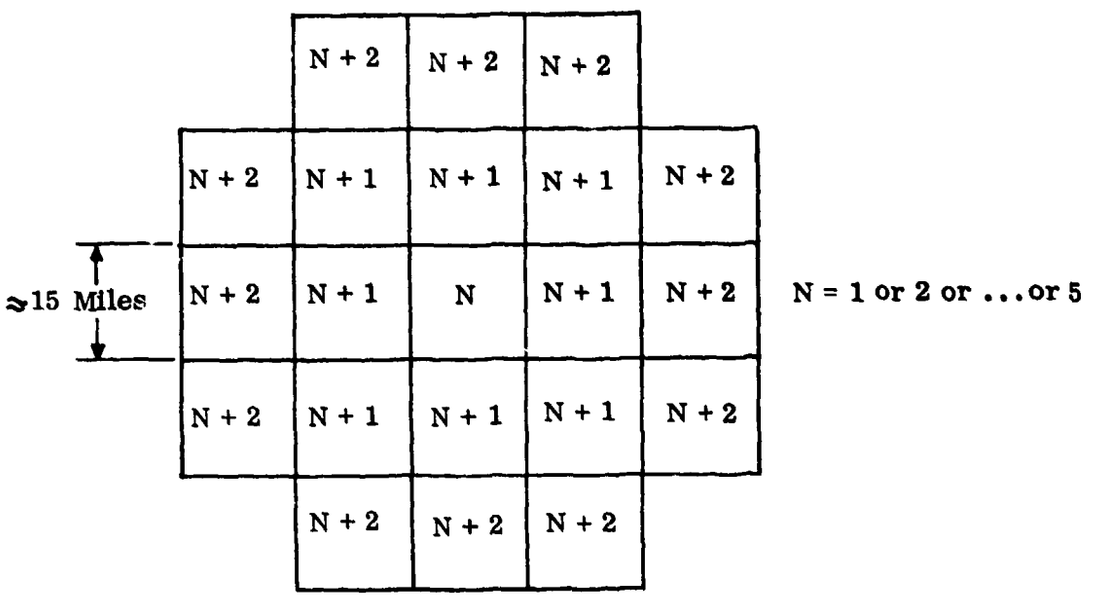


Figure 2.1.4.1-1. Suggested Blocking Scheme

Note: The numbers within the blocks refer to levels of detail. An area of special interest is assumed to lie somewhere within the center block, probably near its middle.

2.1.4.2 Terrain Triangulation Approach

Each terrain surface approximation will involve a different triangulation for each level of detail (LOD). These LOD's may be thought of as resolutions to which the terrain is depicted. A unit decrease in LOD number will correspond to a doubling of the number of triangles in a terrain block's representation. Next, it will be briefly shown how this can be done.

First, an initial triangulation is obtained over a 15' -by- 15N' region. For the sake of concreteness, this very coarse level of detail is labeled "LOD 7" and it is assumed that it contains $2^6 = 64$ triangles. The remaining representations of terrain can be generated in sequence from coarsest to finest LOD. Each triangle of a triangulation, together with a neighboring triangle, forms a quadrilateral which will be called a cluster. Triangles along the borders of blocks, for odd numbered LOD's, will form clusters in themselves since the required neighbor is not within the given block. Finer LOD's are generated by adding a vertex to each cluster of the next higher numbered LOD (see Figure 2.1.4.2-1) and re-triangulating. When generating an even numbered LOD from its odd numbered predecessor of N triangles, the original block of terrain is partitioned into 4 sub-blocks of N/2 triangles each. When generating an odd numbered LOD from its even numbered predecessor of N triangles, the sub-block size is maintained. Thus an odd numbered block of level L will contain twice as many triangles as its predecessor of level L + 1.

As explained above, to obtain each succeedingly lower LOD triangulation, a new vertex must be placed into a quadrilateral. The following criteria in combination provide a good placement rule.

(1) Max-Min Angle Criterion. The minimum angle in the resulting triangulation is to be maximized.

(2) Curvature Criterion. A new vertex is to be placed at a surface specific point; i.e., a point of high curvature such as the peak of a mountain.

If a terrain surface is to be displayed at only one LOD, a complex hierarchical triangulation scheme as described above may not be required. Rather, a set of surface specific points (peaks and pits) and ridge segments may be obtained and triangulated. In this approach, the LOD of the representation will correspond to the number of points and segments over which the triangulation is constructed.

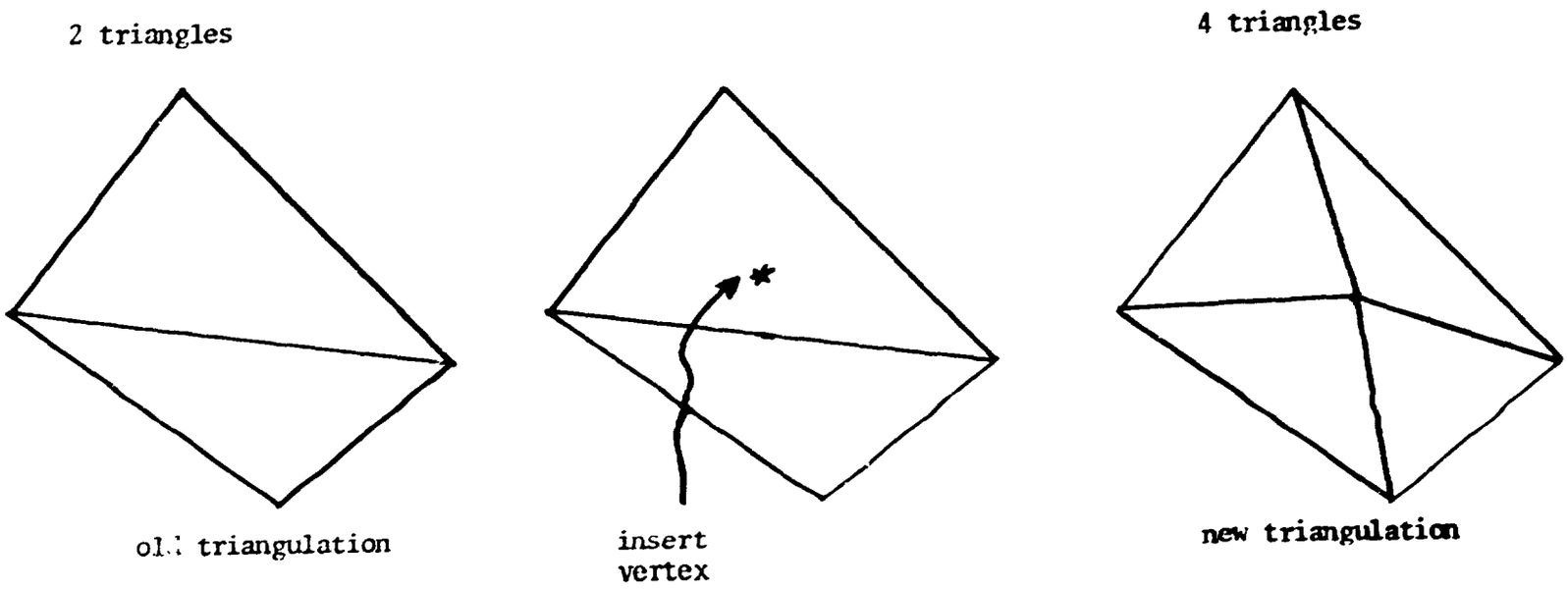


Figure 2.1.4.2-1. Cluster Vertex Addition and Re-triangulation

LOD	Sub-blocks Per 15' x 15N' Region	Triangles Per Sub-block	Clusters Per Sub-block	Total Number Of Triangles Per 15' x 15N' Region	Square Footage Per Triangle	Vertices Per Sub-block	Total Number Of Vertices 15' x 15N' Region
7	1	64	40	64	98×10^6	41	41
6	4	32	16	128	49×10^6	25	100
5	4	64	40	256	24.5×10^6	41	164
4	16	32	16	512	12.3×10^6	25	400
3	16	64	40	1024	6.1×10^6	41	656
2	64	32	16	2048	3.1×10^6	25	1600
1	64	64	40	4096	1.5×10^6	41	2624

Table 2.1.4.2-1. Terrain Representation and Hierarchical Scheme Storage Statistics

2.1.4.3 Storage and Processing Requirements (Example 1)

For this example, the scene content will be limited drastically. The center data block will be set to LOD 2, and three-dimensional objects will only be placed onto that block. The object environment is assumed to consist of 300 generic trees, 100 simple generic buildings, 100 medium detail objects, and 10 very high detail objects. The data requirements per block for this data base is given in Table 2.1.4.3-1.

The storage requirements for disk are as follows:

Non-generic objects	555	32-bit words
Face data	280,000	"
Vertices in face	40,000	"
Unique vertices	48,000	"
Color & texture tables	20,000	"
Total (approx.)	390,000	32-bit words = 1.6×10^6 bytes
		plus 25 percent overhead = 2×10^6 bytes

The above analysis is based upon data storage structures typical of static scene generators (See Table 2.1.4.3-2). Thus, 100 regions of this type can be stored on a single 300 megabyte disk. Disk storage could be reduced by 60 percent and disk access time by 80 percent by storing the data in the form of sequential binary, instead of ASCII. Such a conversion is recommended if disk storage and access time become a problem.

Next, the core storage requirements of this data base will be analyzed. Assuming a 45^o field of view, approximately one LOD 2 block, three LOD 3 blocks, and three LOD 4 blocks which will have to be kept in core, the core storage requirements are as follows:

Generic and non-generic objects	2,550	32-bit words
Face data	116,000	
Vertices in faces	15,000	
Color and texture tables	20,000	
Unique vertices	44,00	
Total (approx.)	200,000	32-bit words

This is a reasonable core size.

Again, assuming a 45^o field of view, the image generation software will be required to process about 5,000 edges. This will take about three minutes using a typical static scene generation software package which is implemented on high-performance 32-bit minicomputer.

	Generic Trees	Generic Buildings	Medium Detail Objects	Very High Detail Objects	Terrain	Culture
LOD 2						
Objects	300	100	100	10	1	
Faces	1200*	800*	800	500	2048	500
Unique Edges	1800*	1000*	1000	800	3000	2500
Unique Vertices	1200*	1000*	1000	500	1600	2500
LOD 3						
Objects						
Faces					1024	400
Unique Edges					1500	1600
Unique Vertices					656	1600
LOD 4						
Objects					512	
Faces					400	
Unique Edges					400	
Unique Vertices					400	

* indicates that these are generic objects for which only one copy need be stored on disk.

Table 2.1.4.3-1. Data Requirements per Block (Example 1)

Entity	Data to be Stored	Storage Required (32-Bit Words)
Object	Number of faces	1/2
	Object priority	1/2
	Centroid	3
	Radius of enclosing sphere	1
Face	Face normal	3
	Face color	1
	Alternate color	1
	Number of vertices	1/2
	Face type (indicates if face is textured, sun-shaded, or special)	1/8
	Texture directions (used only for terrain to obtain smooth transition between terrain faces)	9
Vertices in face	Vertex pointers (assume average of 4 vertices per face)	2
Unique vertices	(x, y, z coordinates)	3

Table 2.1.4.3-2. Storage Requirements Per Entity

2.1.4.4 Storage and Processing Requirements (Example 2)

A second example is now considered using the blocking scheme shown in Figure 2.1.4.4-1, with the block size now set to 7.5' -by- 7.5'. For this example, the contents of each block will be as given in Table 2.1.4.4-1. The storage and processing requirements for this scheme are as follows.

Disk requirements are:

Non-generic objects	9,300	32-bit words
Face data	207,000	"
Vertices in faces	73,000	"
Unique vertices	117,000	"
Tables	20,000	"
Total (approx.)	426,000	words = 1.7 megabyte

Core storage requirements will be derived under a 45^{o} field of view. It will be assumed that one LOD 1 block, and three blocks each of LOD's 2, 3, and 4 will be read into core.

Objects	6,850	
Face data	106,000	
Unique vertices	60,000	
Vertices in face	32,000	
Tables	20,000	
Total (approx.)	224,000	32-bit words

The number of edges that must be processed to produce a scene is 7,900.

These results are roughly equal to those obtained for the last example, under a different blocking scheme.

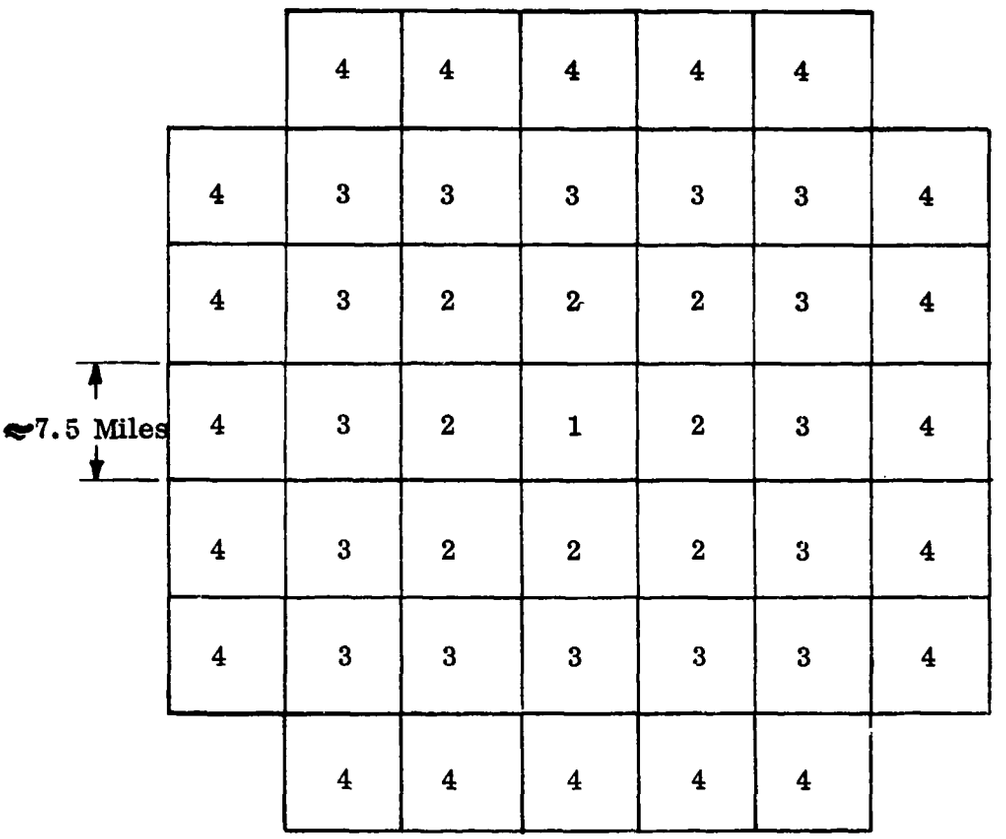


Figure 2.1.4.4-1. Alternate Blocking Scheme

Note: The numbers within the blocks refer to levels of detail. An area of special interest is assumed to lie somewhere within the center block.

	Generic Medium Detail Trees	Generic Low Detail Trees	High Detail Objects	Low Detail Objects	High Detail Buildings	Low Detail Buildings	Terrain	Culture
LOD 1								
Objects	100	100	20	100	50	100		
Faces	1600	400	640	800	1600	600	1024	400
Unique Edges	3200	600	1280	1200	3200	1200	1500	2000
Unique Vertices	2000	400	800	800	2000	800	656	2000
LOD 2								
Objects		100		100		100		
Faces		400		800		600	512	200
Unique Edges		600		1200		1200	600	800
Unique Vertices		400		800		800	400	800
LOD 3								
Faces							256	150
Unique Edges							400	300
Unique Vertices							164	300
LOD 4								
Faces							128	
Unique Edges							180	
Unique Vertices							100	

Table 2.1.4.4-1. Data Requirements per Block (Example 2)

2.2 TECHNOLOGY REVIEW

Sensor scene simulation is closely related to computer graphics, CIG, and image processing in the following sense. It is a special application of computer graphics because perspective displays of computer modelled data are generated. It is related to CIG because flight training applications of sensor scene simulation require realtime image generation. Furthermore, the data structures and content of scenes displayed in sensor simulation are very similar to those in visual simulation. Image processing is relevant because many sensor transfer functions must be modelled by local image processing mechanisms such as filtering and because source data is often derived from processed images. The following paragraphs discuss the three fields and their relation to sensor simulation.

2.2.1 COMPUTER GRAPHICS AND SENSOR SCENE SIMULATION

Sensor scene simulation is a specialized application of computer graphics in the following sense. A geometric abstraction of a real world environment is mathematically modelled for the purpose of generating graphics images. A wide variety of images may be generated from the same data by varying the pilot's viewpoint or varying display attributes and processing to mimic the properties of a variety of sensors.

Computer graphics provides a rich source of tools for representing, storing, and rendering such images. Tradeoffs between algorithms, data structures and hardware in computer graphics provide useful guidelines for sensor simulation. The following is a discussion of key issues in computer graphics concerning display processing algorithms.

2.2.1.1 Computer Graphics Algorithms

The most general and realistic approach to scene generation is raster scan display. A shaded monochromatic or color picture is specified in terms of the shade or color at each point of the image. The scene is usually displayed on a TV screen.

A raster image is generated not by tracing out sequences of geometric items, such as lines or characters, but by "painting" out colors on a point-by-point basis. Complex scenes result from complex patterns of color and illumination. Thus, the color of every pixel (picture cell) in the raster matrix must be determined before the image can be displayed. This involves the conversion of a geometric data base to a raster scan image, a process called raster scan conversion.

To create the illusion that the interior of a displayed surface is opaque requires that the parts of the scene hidden from the observer not be displayed. The task of displaying a scene with proper occulting is called hidden surface elimination.

Terrain scenes can be modelled by a triangular faceted surface approximating the topography. Convex three-dimensional objects can be placed on this piece-wise planar terrain surface. Since it is stipulated that objects be plane-faced, a simple face normal calculation is sufficient to determine which faces are potentially visible (i.e., front facing). Other simplifications are possible if objects are not allowed to pass through each other or overlap cyclically (Figure 2.2.1.1-1).

Several approaches that have been used to generate the type of scenes that a pilot sees out of the window during low level flight are reviewed below.

2.2.1.1.1 List Priority Algorithms

The list priority approach was developed by the General Electric Company in the 1960's for flight simulation. It has been implemented in hardware by the General Electric Company and Singer/Link Company and in software by General Electric Company, Evans and Sutherland Company, and Encarnacao, et al., at the University of Minnesota. See references C.23 and G.14 for further details. This approach to scene display is reviewed below.

a) Triangular Surface Topography

If a viewray strikes but a single terrain face, the viewray will be assigned the color of the face. If the viewray strikes two or more terrain faces, the viewray will be assigned the color of the terrain face whose centroid lies closest to the viewpoint. This procedure will fail if a point x on a terrain triangle T is closer to the centroid of some non-adjacent triangle than it is to the centroid of T (see Figure 2.2.1.1.1-1. This situation can be tested for and eliminated during data base development by assigning an ad hoc distance which satisfies desired distance relationships.

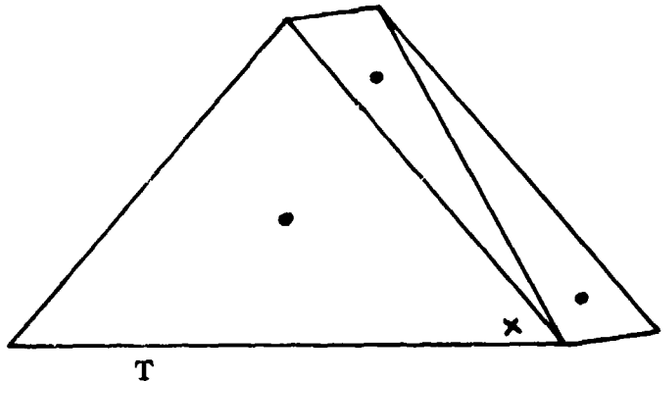


Figure 2.2.1.1.1-1. Priority for Terrain Triangles

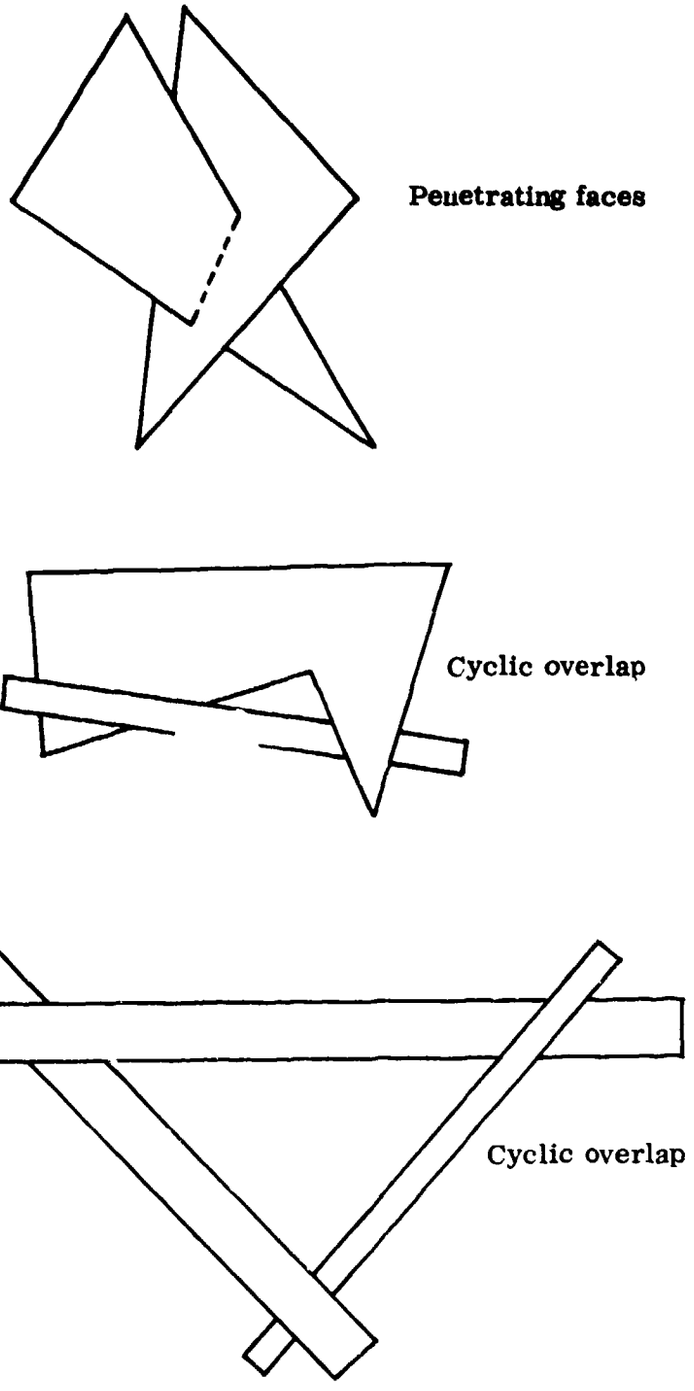


Figure 2.2.1.1-1. Forbidden Face Relationships

b) Ground Material Representation

Ground material may be represented by placing colored polygons over terrain polygons, assigning a higher priority number to each colored polygon than the terrain polygon below it. These may be overlaid to preserve edges; when the viewray strikes several such polygons residing on a single terrain triangle, it will be assigned the color of the polygon having highest priority.

Now suppose that a viewray strikes a number of colored polygons residing on different terrain triangles. First, it will be determined which terrain triangle has precedent, by distance to centroid. Then the viewray will be assigned the color of the face with highest priority lying on the triangle.

This scheme for displaying such ground material color, is foolproof as long as the triangular terrain surface approximates a single-valued surface.

c) Three Dimensional Objects

The solution for 3D objects is somewhat more complicated, but can be greatly simplified by grouping objects into clusters.

An object can be defined as a collection of convex faces enclosing a single convex volume polyhedron. The face priority calculations for a cluster of objects can be made during data base development independent of viewpoint. A cluster can be defined as a collection of faces which can be assigned viewpoint independent priority numbers, such that after back faces are eliminated from consideration, the correct priority for any face in the cluster can be determined from any viewpoint.

A cluster should be placed on one and only one terrain triangle to take advantage of the simplicity of terrain polygon priority calculation. No part of the cluster should be allowed to extend out beyond the vertical walls which could be formed by constructing planes perpendicular to the base triangle, passing through its edges.

Any cluster will have higher priority than the face upon which it resides. If a viewray strikes several faces of a cluster and the triangle upon which it resides, it will be assigned the color of the front facing face of highest priority.

Now suppose the viewray strikes a cluster and a front facing triangular face F upon which it doesn't reside. The viewray will be given the color of triangle F if the triangle's centroid is closer to the viewpoint than that of the triangle upon which the cluster resides, and vice-versa.

Separation planes can be used to separate clusters which reside on the same terrain polygon. This avoids the calculation of priority between individual objects of each cluster. This method is illustrated in Figure 2.2.1.2.1-3.

d) Raster Scan Conversion

The basic modules required to display an image using the list priority approach are shown in Figure 2.2.1.1.1-2. The operation of each module is described below.

Perspective Projector - The Perspective Projector selects those edges from the original 3D data base which appear within the field of view (a clipping operation). The perspective coordinates of each edge in the environment are computed, giving edge equations of the form

$$J = a + bI.$$

This module also adjusts face colors as a function of sun/moon azimuth and elevation.

Priority Processor - The Priority Processor uses viewpoint, separation plane data, centroid locations, and relative priority numbers to compute a unique absolute priority number for each face in the environment. The faces are listed in priority order, with back facing faces excluded.

Edge Generator - The Edge Generator receives edge data from the Perspective Projector. It stores potentially visible edges. It calculates the line intercept J-values for the two endpoints of each potentially visible edge. The Edge Generator determines which edges intersect the top and bottom of each scanline.

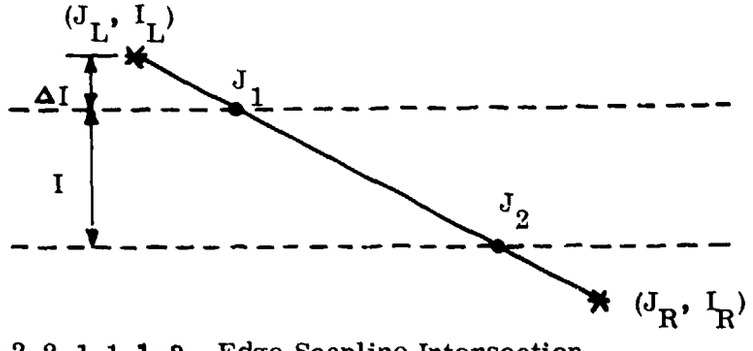


Figure 2.2.1.1.1-2. Edge-Scanline Intersection

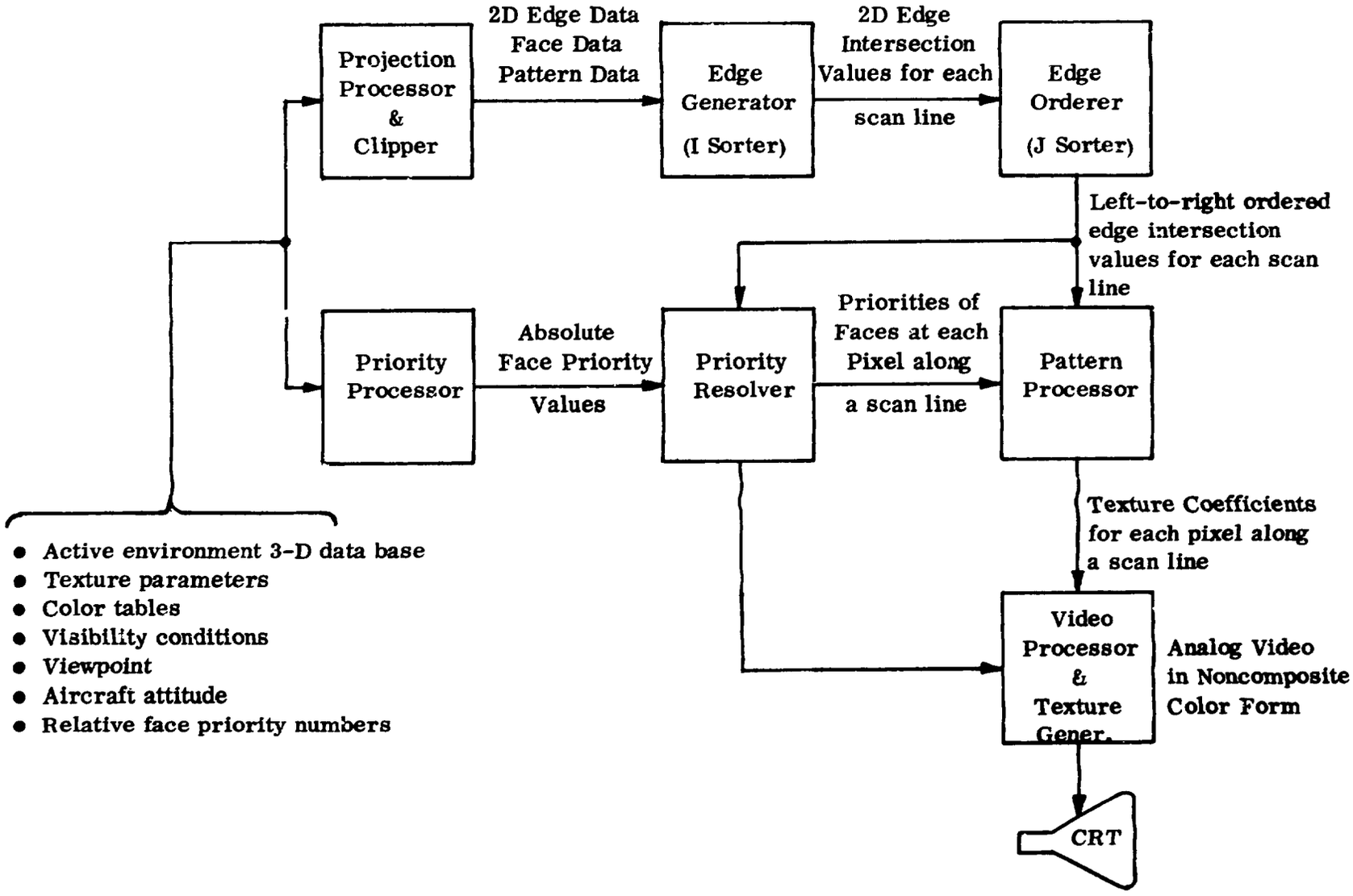


Figure 2.2.1.1.1-3. Raster Scan Conversion Processing

$$SLOPE = (J_1 - J_L) / \Delta I$$

$$J_1 = J_L + \left|SLOPE\right|\left|\Delta I\right| \; ; \; J_2 = J_1 + \left|SLOPE\right|$$

Thus the second intercept value can be expressed as the sum of the first value plus an incremental term, and so on until the lower endpoint of the edge is reached.

Edge Orderer - The Edge Orderer receives intercept values and edge words from the Edge Generator. The Edge Orderer orders the potentially visible edges by J-value, for each scan line.

Priority Resolver - The Priority Resolver receives potentially visible face data from the Priority Processor. The Priority Resolver creates a face priority list in which face priorities are addressed by face number. For each raster line, the Priority Resolver receives potentially visible edge data from the Edge Orderer. For each edge, the edge's face number is used to obtain from the face priority list the relative priority number of the edge.

If an area times color edge smoothing technique is used, several faces contribute to the color of each pixel. If an oversampling approach is used, only one face is considered as potentially visible at each subpixel.

Visible edge words are sent to the Video Processor.

Pattern Processor - For each scan line, the Pattern Processor receives J_L and J_R values for each scanned face along with the face number. Texture data is obtained for each face and texture coefficients are computed.

Video Processor and Texture Generator - The Video Processor receives visible edge words from the Priority Resolver and texture data from the Pattern Processor. A set of red, green, and blue intensity values is computed for each pixel based upon the weighted colors of the faces which project onto the pixel. This color is modified by texture values and a surface fading function.

2.2.1.1.2 Scanline Coherence Algorithms

a) Introduction

Scanline algorithms solve the hidden surface problem one video raster line at a time. Several approaches, summarized below, are described in references G.28, G.36, G.3 and G.20. When faces in 3D space are projected onto the view window, several faces may superimpose in projection. The intersection of a scanline and a projected face is called a segment. The horizontal endpoint coordinates of a segment may be computed as an incremental function of the scanline number I.

Each scanline may have a number of segments across it (Figure 2.2.1.1.2-2). Segments may overlap. The depth relationships between segments in the I-J plane determine visibility.

A section of a scanline which contains no segment endpoints will be called a span. The spans of a scanline are identified by sorting all segment endpoints by J and examining regions delimited by these J values. A span is a section of a scanline in which no changes of visibility can occur. A sample span must be tested for visibility if: (1) the number of segments on the current scanline differs from the number of segments on the previous scanline, or (2) if the associated of segment endpoints with the edges of the projected faces is interchanged when proceeding from the previous to the current scanline. In this case the two edges intersect between the two scanlines.

Most algorithms exploit the coherence properties of scenes to reduce the computational load. That is, geometric situations which do not change from frame to frame or line to line are not recomputed. The process of forming the segments of a scanline requires the calculation of the intersection between the scanline and the edges of the projected surfaces. The required computations can be greatly reduced if the objects have been previously sorted according to their maximal I coordinate. Much of the remainder of the workload involves sorting in the J and K directions. A single Active Edge List, sorted by J, will contain edge information for all polygons which intersect a scanplane. This list is resorted for each new scanline processed (possibly only if there are changes in order).

An Active Polygon List will be updated as each scanline is processed from left to right. Each projected polygonal edge intersected by a scanline may be described as either a left or right edge (Figure 2.2.1.1.2-2). As the left edge is scanned the polygon becomes active, and remains so until the right edge is reached. This left-right nature of edges can be characterized by either (1) associating a single bit with each polygon to record whether or not it is active, and by complementing the bit each time an edge of the polygon is encountered; or (2) if polygons are always drawn in a consistent way (e.g., counterclockwise when viewed from outside the object), by letting edge direction determine parity (e.g., a downward pointing edge is a left edge).

Algorithms may be designed which take different approaches to (1) selecting spans, (2) taking advantage of coherence between adjacent scanlines, and (3) taking advantage of coherence within a scanline. The Watkins algorithm (G.36) is the most popular of the group for both software and hardware implementation. Programs using this algorithm are readily available (G.10, G.26). Hamlin and Gear (G.20) demonstrate that their CROSS algorithm is much faster than any of the other three (at least for their set of test scenes). However, none of these algorithms should be viewed as set in concrete. Clever ideas found in one algorithm can be used in any of the others.

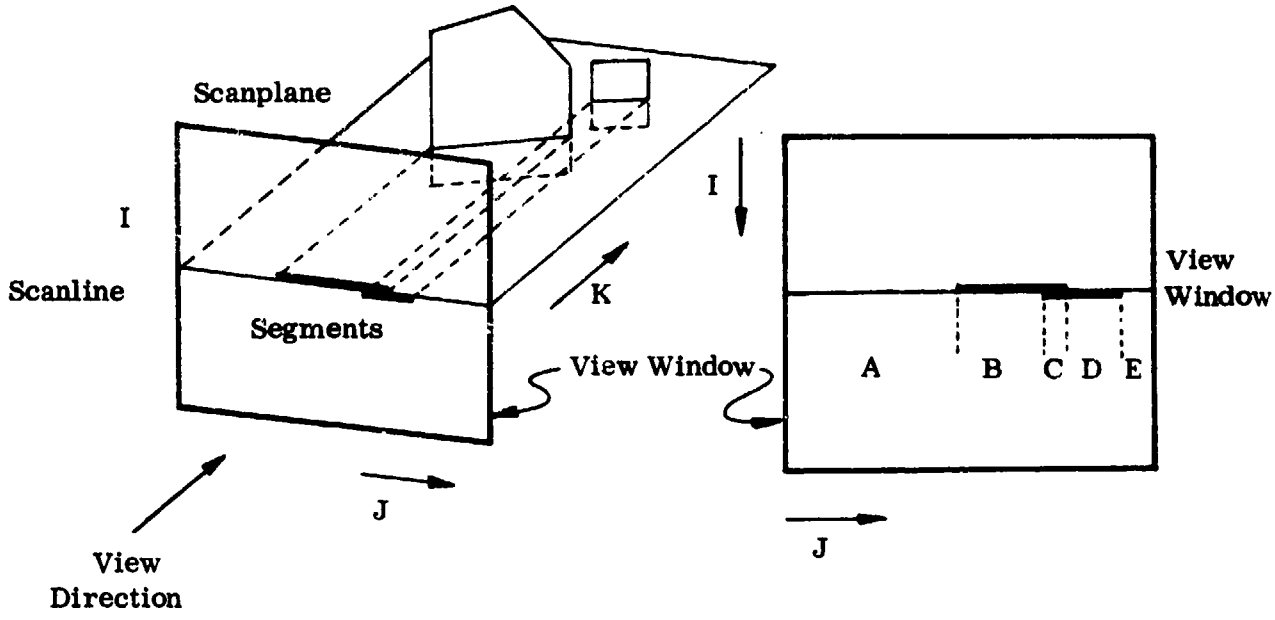


Figure 2.2.1.1.2-1. Scanplane Projection to View Window

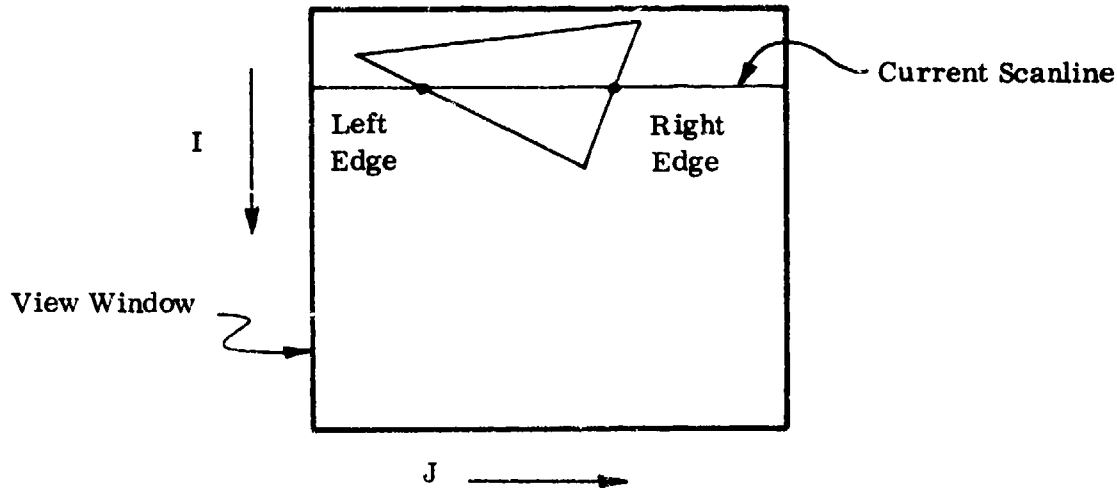


Figure 2.2.1.1.2-2. Polygon-Scanline Intersection

b) Operation

The basic modules comprising a SCANLINE COHERENCE algorithm are briefly reviewed below.

Projection and Clipping - This is essentially the same as for the LIST PRIORITY algorithm.

I-sort - All the algorithms bucket sort by the I-coordinate of one of the vertices of an edge. Romney's algorithm (G.28) keeps an I-Occupied Table to record which faces intersect a scanline.

J-Sort and Merge - Romney's algorithm bucket sorts the intercepts of all projected faces that intersect a scanline. A J-Occupied Table records which faces are potentially visible for each raster element. This algorithm recomputes the depths of all faces whenever a face enters or leaves the J-Occupied Table.

The other algorithms take special advantage of the J-intercepts of edge crossings. These algorithms use a J-Sort List in an analogous manner to Romney's use of a J-Occupied Table. When a new scanline is encountered, the J-sorted edges that enter on the scanline are merged into the J-Sort list. Any edges in the J-Sort List that exit on the new scanline are deleted. Then the list is re-sorted in J value by a bubble sort. The bubble sort is extremely fast in this situation since very few edges cross each other from one scanline to the next.

Span Extraction, Culling, and Visibility Calculation - First, a scanline must be partitioned into spans. Within each sample span, the edges that fall within the span must be culled out and examined. This involves searching or sorting the segments that fall within a span to determine which one is visible.

Romney's algorithm monitors the faces entering and leaving the J-Occupied Table. When such a condition occurs, the algorithm recomputes the depths of all faces in the table to determine which one is closest to the viewpoint. This decision persists until the next change to the J-Occupied Table. Sample spans are determined by edge crossings.

Romney takes advantage of the fact that faces are not allowed to penetrate each other. He observes that if exactly the same faces are present in one scanline as in the previous scanline, and if the J-ordering of the edge crossings is exactly the same, then the computed depth values are still valid. The same faces that were visible on the previous scanline will still be visible, although their extents in J may differ.

Romney's algorithm does not take advantage of the coherence of J intercept values from one scanline to the next.

In Bouknight's algorithm (G.2), edge crossings define the limits of sample spans. As each new span is entered, a new depth computation is performed to decide which face is visible within the span. The algorithm uses a one bit face code to indicate whether a face falls within the current span.

Watkins' algorithm (G.36) generates spans more aggressively than the previous two algorithms. In this algorithm, the left end of a sample span is fixed, and the right end "floats". Initially, the right end coincides with the right end of the scanline. As new segments are extracted from the J-Sort List, the right end of the sample span may be moved to the left until the situation represented within the sample span is simple enough to allow for the easy computation of which edge is visible.

The Watkins algorithm uses a very quick logarithmic depth search. Rather than solving exactly the plane equations of all faces that project into a sample span in order to determine their depth, the algorithm computes only as much information about the depth as is required to determine which face is visible.

Hamlin and Gear's algorithm called CROSS (G.20) also exploits the fact that if it is assumed that polygons cannot intersect, then the relative depth of segments cannot change, and the only times that it is necessary to perform new depth calculations are when the order in the Active Edge List changes. Their algorithm uses a form of object coherence across edges to infer the visibility of a face connected to an edge from that of an adjoining face whose visibility has already been computed. That is, when the order in the Active Edge List changes, the visibility of polygons after the change can often be deduced from the visibility before the change and an examination of the type of edges involved in the change.

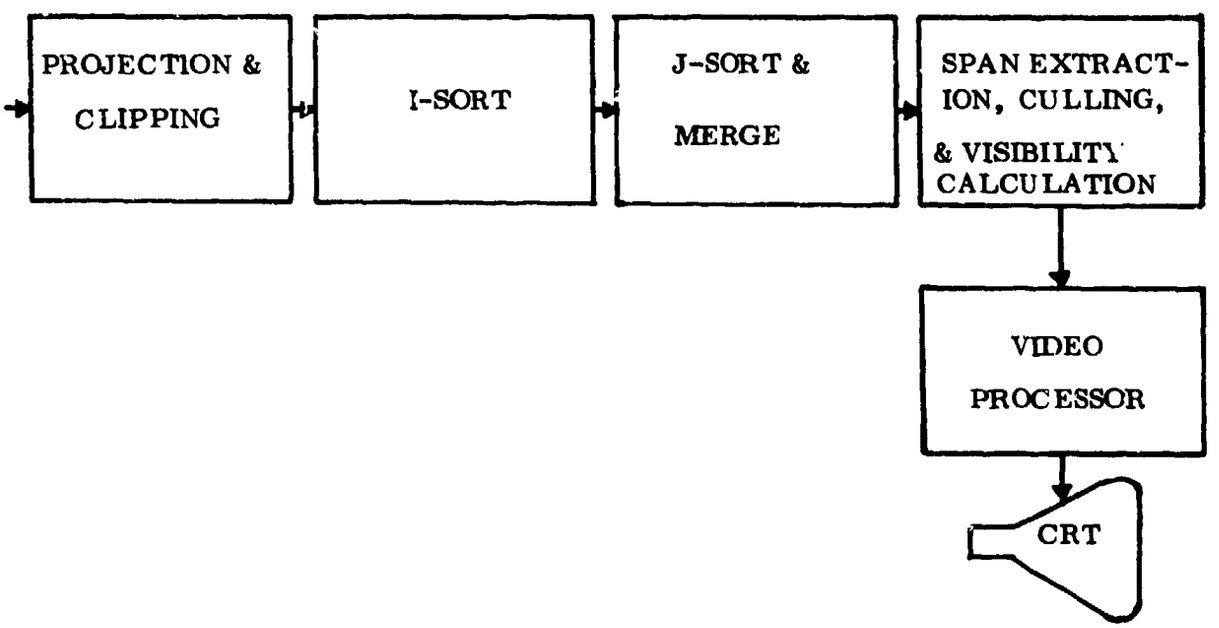


Figure 2.2.1.1.2-3. Generic Scanline Coherence Processing

2.2.1.1.3 Polygon Span Algorithm

The Polygon Span Algorithm is a two-dimensional version of the previous class of algorithms. It operates as follows. First, all front facing faces are projected onto the view window. Then span polygons are formed in the projection space, such that no edges cross a span polygon. These span polygons partition the view window into tiles. The aggregate of tiles forms a mosaic image. If a span polygon represents several projected faces, the corresponding tile is assigned the color of the face closest to the viewpoint. An image is produced by scanning out the mosaic one tile at a time. Each tile is scanned out by moving the electron beam back and forth between its boundary. Tiles may have to be partitioned into simpler tiles, so that the electron beam need not be blanked out during their scan.

This approach is not well-suited for sensor simulation for several reasons. Most importantly, the number of computations required to determine the span of polygons becomes prohibitive as the scene becomes complex. Sensor simulation scenes are typically more complex than artificial graphics laboratory scenes. This approach is best implemented with a color calligraphic CRT, i.e., a system with a ramdomly steerable electron beam. Since such devices typically use beam penetration tubes, the number of different colors which can be displayed is very limited. It would be also very difficult to display texture in perspective with this approach.

2.2.1.2 CIG and Sensor Scene Simulation

The most advanced techniques in computer graphics over the last decade have been used to generate extremely rich artificial images which include such features as curved surfaces, translucent refracting surfaces, rich surface textures and colors, and extremely complex models (G.26). Many of these techniques have been developed in academic environments with the goal of producing the richest possible images often at the expense of processing time and storage. In sensor simulation, the goals are rather different. Image quality must match that of rather limited sensors, scene content is much more complex, and processing efficiency is important from a cost-effectiveness point of view. This situation also characterizes visual flight simulation where image quality requirements always play a secondary role relative to processing speed requirements. Data base structure and content are similar in sensor and visual simulation; furthermore, sensor simulation in software often incorporates the same algorithms used in CIG hardware systems. For these reasons the paragraphs below discuss the major processing approaches in the design of the world's most advanced flight simulators. CIG has provided a crucible for weeding out inefficient processing from the gamut of all possible algorithms found in non-real time computer graphics. The surviving algorithms and system designs are often worthy candidates for software emulation in sensor simulation research.

The CIG systems described below are:

- Singer-Link Digital Image Generator
- General Electric ASUPT
- Gould GVS-1
- McDonnell Douglas VITAL IV
- General Electric NASA
- Advanced Technology Systems COMPUTROL
- Evans and Sutherland CT-5
- Marconi Radar

2.2.1.2.1 Singer Link Digital Image Generation System

The 32-bit general purpose computer at the front end of the Singer system (Figure 2.2.1.2.1-1) is the Digital Image Generation Controller (DIGC). It has 384K of private core and supports the usual peripherals. The DIGC accepts ownship heading, position, attitude, rate of attitude change, velocity, weather conditions, and other data from the flight computer. Based on these inputs, it dynamically extracts object descriptions from disk and places them in an Active Data Base (ADB) memory. The ADB is that portion of the total data base which is within the vicinity of the aircraft at any point in time. The DIGC also computes occultation priorities, controls the allocation of ADB memory, and directs the activities of the image generation hardware (C.21).

The DIGC's software modules fall into three categories: extrapolation, synchronous, and asynchronous. Extrapolation modules execute as high priority foreground tasks to determine the position and attitude of owncraft and all moving scene components each frame time. This requires an extrapolation of the input data which are not synched to the image generator's frame rate. Synchronous modules compute other data as required by the image generation hardware. Asynchrounous modules run as background tasks performing less urgent functions such as data retrieval and priority calculations.

The data base is organized into 80 x 80 nautical mile regions, each partioned into 225 5.4 x 5.4 mile "area" blocks.

Data is retrieved from disk in the form of either area records or cluster records. An area record contains no object descriptions, but rather identifies the clusters of objects within an area block.

The DIGC identifies the area block below owncraft and retrieves from disk all area block records within a square centered about this particular block. Singer calls this square region a panorama (Figure 2.2.1.2.1-2). The chosen panorama size is a function of mission visibility conditions.

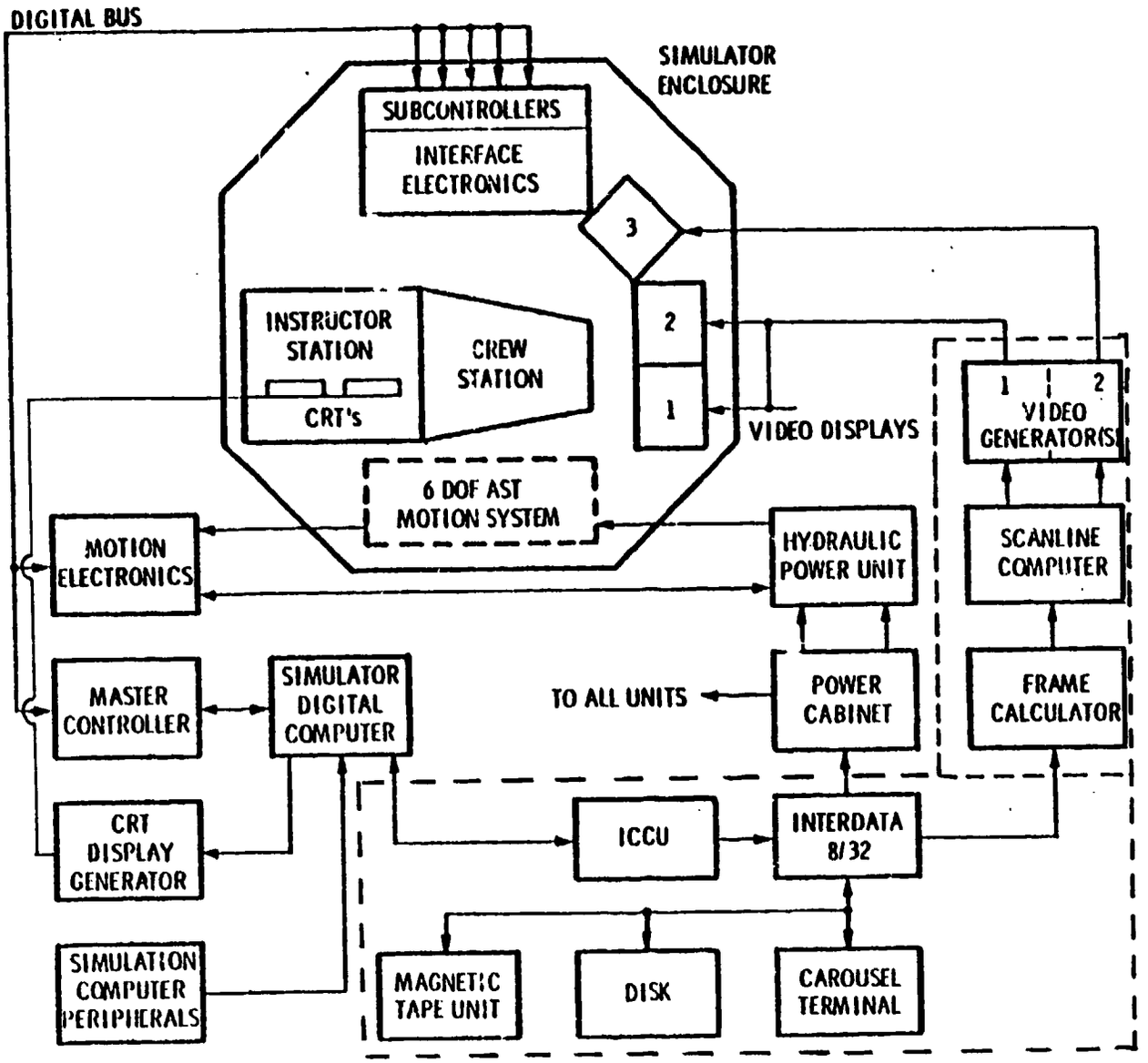


Figure 2.2.1.2.1-1. Singer Link Flight Simulator System Diagram

Each cluster in the panorama is tested to determine if any portion of it is potentially visible. A pyramid of vision slightly larger than the FOV is used to prevent lags during attitude changes. A code is assigned to each cluster to indicate if it is in view, out of view, or partially in view.

The level of detail at which a cluster is retrieved is based upon the distance between the viewpoint and the cluster's centroid, and switching range values stored with the cluster. By allowing switching ranges for successive levels of detail to overlap, frequent switching between two levels of detail is prevented.

The priority list approach is used to solve the hidden surface problem. Certain priorities are computed in advance and stored on the Visual Data Base disk. Separation plane data is also stored on the disk in binary tree form (Figure 2.2.1.2.1-3b). The nodes of the tree represent separating planes, and the leaves convex objects. The relative priorities of objects that have not been previously specified are determined by the DIGC on a block by block basis. Relative object priorities within a block are determined by searching the separating plane tree. This approach is also used for airborne clusters by placing a separating plane below the cluster. Finally, relative priorities are converted into absolute priorities by using the relationships among the area blocks. This process uses the observation that an object in a ar block cannot obscure an object in a nearer block (Figure 2.2.1.2.1-4).

The DIGC is connected to the image generation hardware through three direct memory access (DMA) channels. The ADB is transferred to the ADB memory through one of these channels. Geometric processing is under software control through the second channel. For every frame, the position and attutude of ownship and all moving objects is transferred to a Frame Calculator, followed by a list of objects to be processed for each video channel. The third DMA channel is used to transfer color codes.

The Frame Calculator is arranged into a pipeline composed of an Object Processor, Face Visibility Tester, Rotation and Illumination Subsystem, Clipper and Face Boundary Calculator (Figure 2.2.1.2.1-5). The Object Processor examines every object in the list received from the DIGC and eliminates all backward facing faces and all faces which are very small in projection. Part of the Object Processor's task is to transform pilot position into object coordinate systems. The resultant vertices are passed to the Rotation and Illumination Subsystem together with a list of front facing faces. The Rotation and Illumination Subsystem transforms vertices of faces into window coordinates and in parallel determines the illumination for each face. This data passes on to the Clipping Subsystem which uses a modified version of Sutherland's clipping algorithm. The Boundary Calculator receives sequences of vertices defining faces. It calculates and outputs all necessary display parameters in view plane coordinates. The first section of the Boundary Calculator is a pipeline processor that projects vertices onto the view windows. Divisions are performed in the logarithm domain, using a logarithm look-up table, a subtractor, and antilog look-up table. The second section of the Boundary Calculator is a parallel group of processors that convert vertices into view

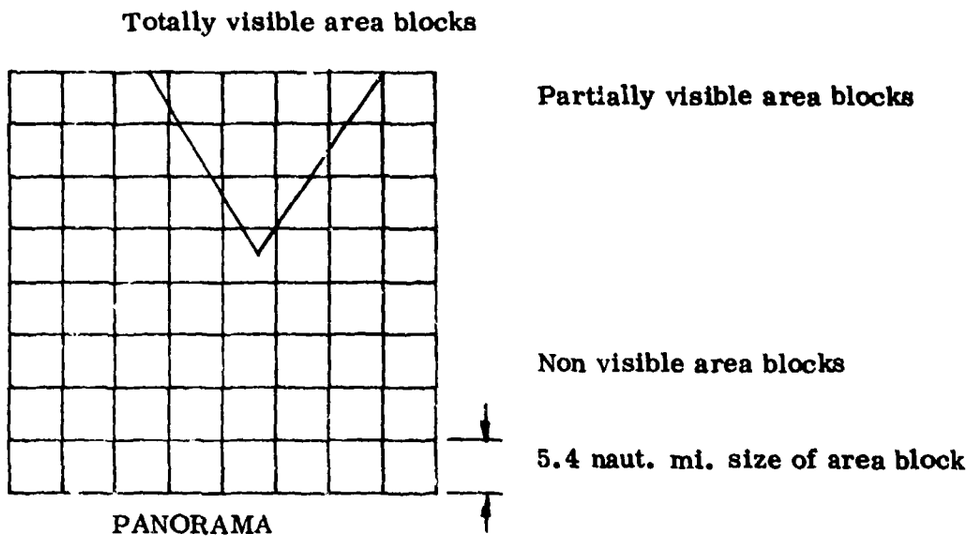


Figure 2.2.1.2.1-2. Potentially Visible Area Blocks

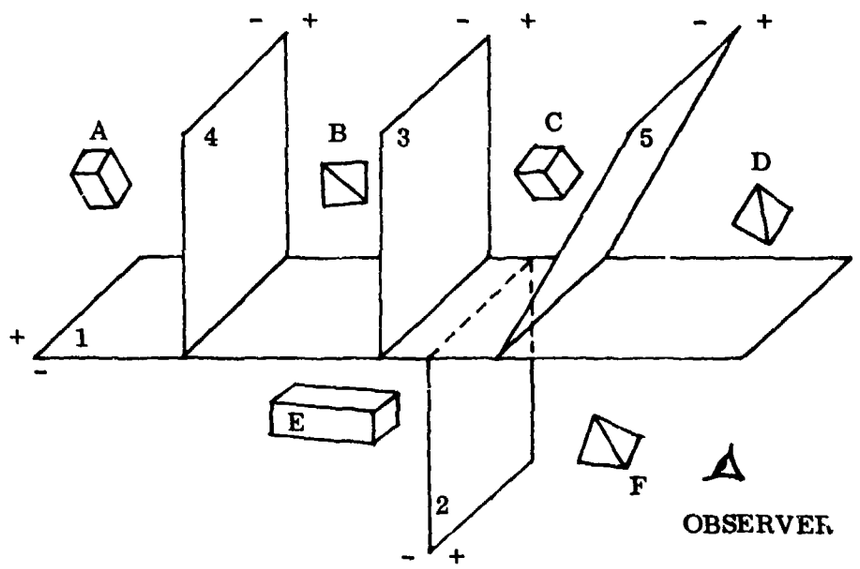


Figure 2.2.1.2.1-3a. Separating Planes

Note: ("+" and "–" signs refer to the dot products of the view vector with surface normals)

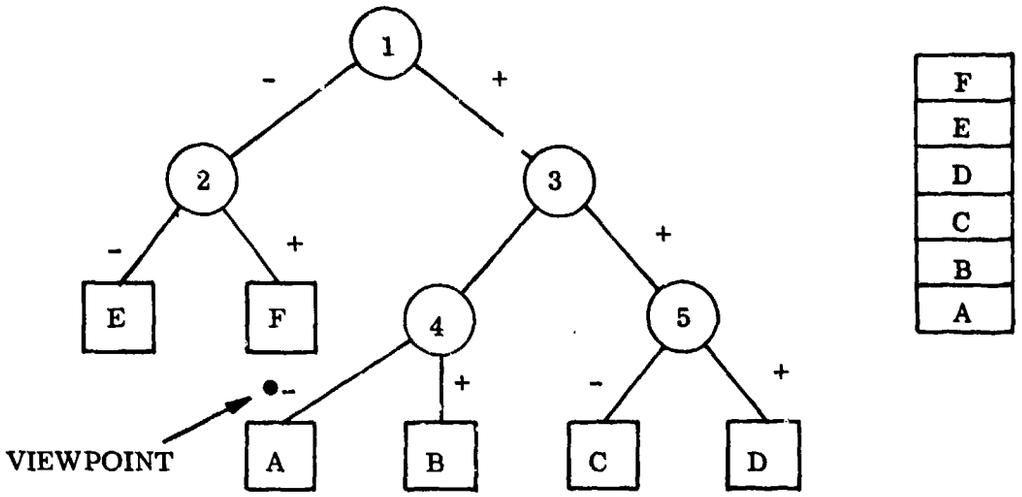


Figure 2.2.1.2.1-3b. Separating Plane Tree Node Priority List

6	5	4	3	4
	4	3	2	3
	3	2	1	
		1	0	

Figure 2.2.1.2.1-4. Relative Area Block Priorities

Note: ("0" denotes the highest priority).

plane edges. Edge data consists of the (I, J) locations, intensities, and fading values at the end points of each edge. The delta of these values is also computed. The third section of the Boundary Calculator is a pipeline processor which computes the rate of change (in terms of elements per scan line) of J coordinates, intensity, and fading coefficients. Divisions are carried out in the logarithm domain.

Between the Frame Calculator and Scanline Computer is a double buffer memory to hold two edge lists. Geometric data produced by the Frame Calculator is written into one edge buffer. At the same time, data produced by the Frame Calculator during the previous frame time is read out of the other edge buffer and into the Scanline Computer. When the Frame Calculator and Scanline Computer have completed processing a frame's worth of data, the two sections of memory are switched for the next frame time's processing. This type of memory switching is often referred to as double buffering or "ping-ponging".

The Scanline Computer (Figure 2.2.1.2.1-6) consists of a Sort and Update Subsystem, Occulting Logic, and Video Generator arranged into a pipeline.

Data coming out of the edge buffer memory passes through the update logic. There, the starting parameters are updated, according to which of the interlaced fields are being processed. The updated edge data enters the sort logic where up to 256 edges per scanline are ordered by J-value. The sort is accomplished by two passes through a pipeline. The sorted J-values along with other edge data are sent to the Occulting Subsystem. Sorted edge data is also fed back to the Update Logic where it is used in the next field's processing.

The Occulting Logic computes the visibility of intersections and determines the intensity of partially obscured surfaces at the point which they become visible. Visibility is determined by examining edge intersections on a scan line, in sequence, and creating an "active object" list. An object is placed into the list when its leftmost edge is encountered and removed when its rightmost edge is reached. A temporary memory holds the intensity associated with the last encountered intersection of an object, along with the intensity increment and color to its right. An intersection point is visible if it results from an object whose priority is the highest in the active object list. If the intersection point results from a beginning or internal edge, the intensity at that point is determined by the parameter of that edge. If an intersection point results from an ending edge of a visible object, the intensity assignment is taken for the next highest priority object in the active object list (meaning that the left portion of this next highest priority object is obscured). The last encountered edge parameters of this partially obscured object are then retrieved from the temporary memory. The correct intensity value is computed for the particular location along the scanline at which the partially obscured object becomes visible.

The output of the Occulting Logic is a compressed description of the color along a raster line. It is in the form of a sequence of visible intersections, their associated intensity values, the intensity increments for the faces to their right, and color codes.

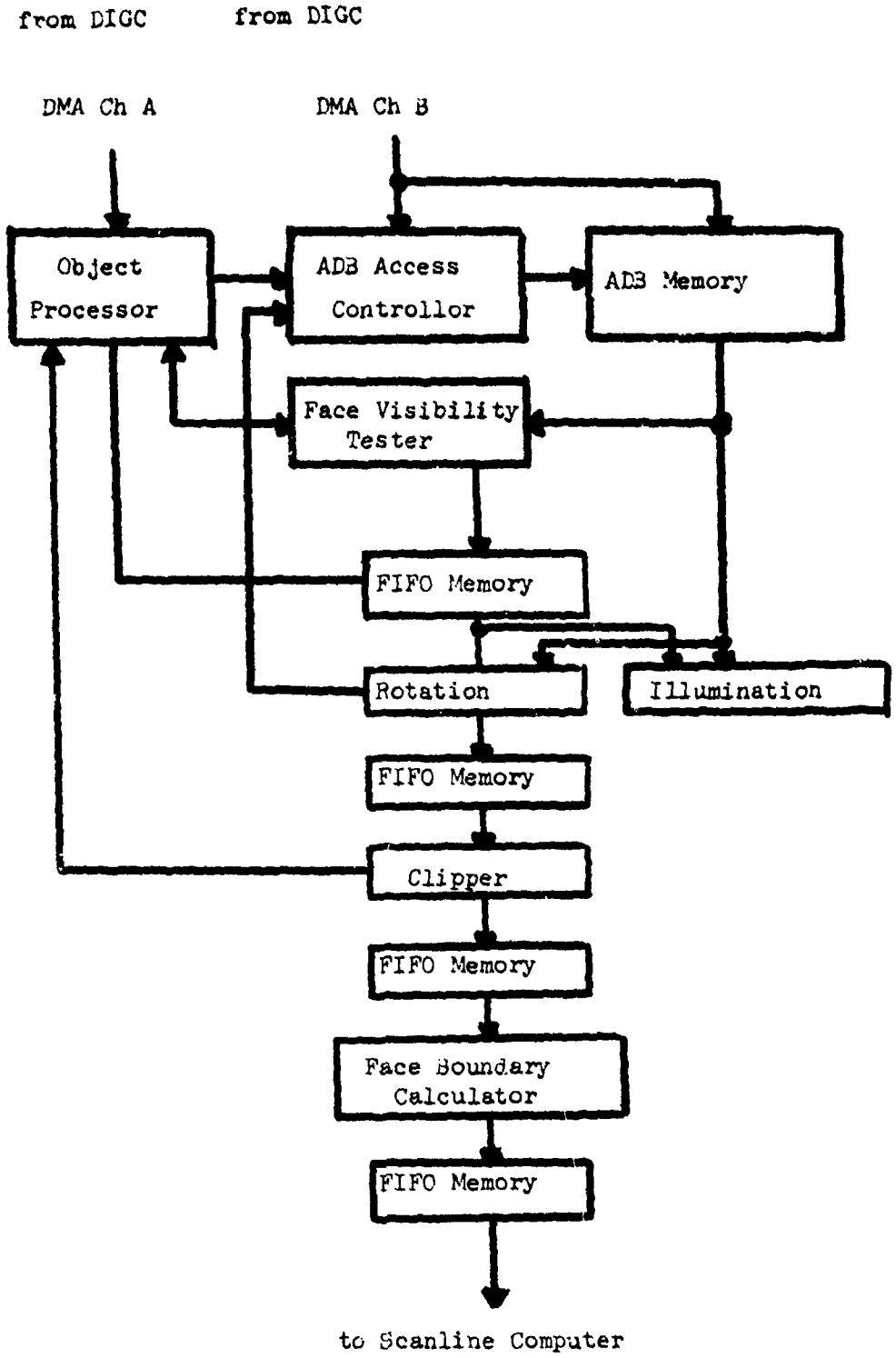


Figure 2.2.1.2.1-5. Frame Calculator Block Diagram

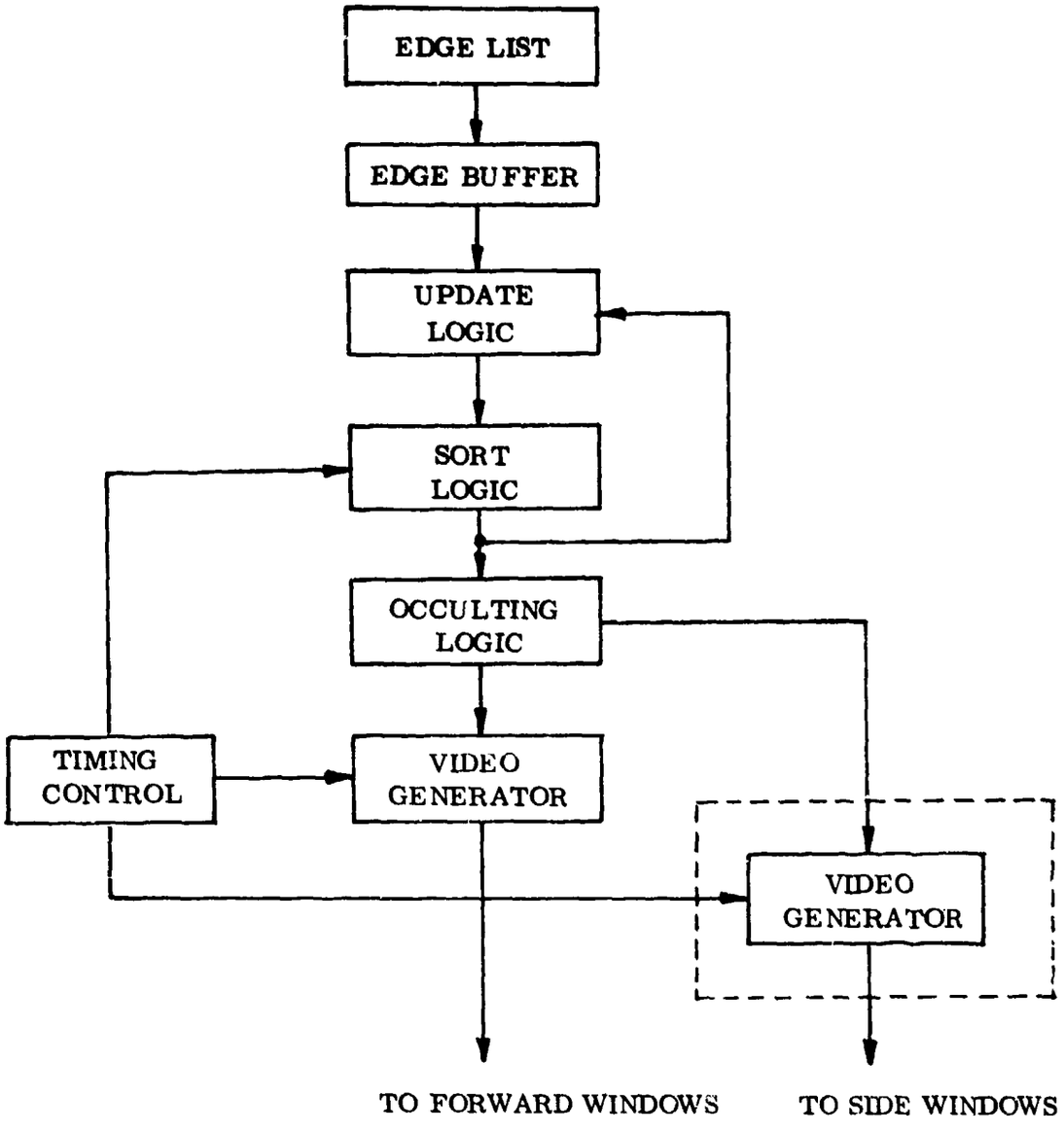


Figure 2.2.1.2.1-6. Scanline Computer Block Diagram

The Scanline Computer processes the data for each scanline serially, one channel at a time. The processed data for one scanline of each channel is stored in a buffer memory. This memory permits simultaneous scanline output for each channel to separate Video Generators. At the same time, data for the next scanline is being computed and stored (Figure 2.2.1.2.1-7).

A Video Generator (Figure 2.2.1.2.1-7) expands the compressed description of a scanline into 8 bits each of red, green, and blue values for each point along a raster line. This is done by using a look-up table to convert the color codes into color components. An intensity value and red, green, and blue color components are fed into four fast D/A converters to provide three video signals. Finally, these red, green, and blue signals are modified by fading, and horizontal and vertical edge smoothing operations.

2.2.1.2.2 General Electric ASUPT* System

The ASUPT (Advanced Simulator for Undergraduate Pilot Training, Williams AFB, Arizona) system consists of three major hardware units (Figure 2.2.1.2.2-1); Frame I: general purpose computers, Frame II: special purpose computer, and Frame III: raster scan conversion and CRT electronics. The three computation units are connected serially to form a pipeline processor with 3/30 of a second throughput delay. (C.1).

Frame I is built around two general purpose central processor units (CPU A and CPU B). CPU A performs arithmetic operations and "talks" to the flight computer. It has 8K of core and access to 8K of core in the flight computer. CPU B handles certain management tasks including the search and retrieval of data blocks from the Visual Data Base Disk, to describe each new geographic region the pilot encounters. CPU B has 8K of core to itself and 2 8K modules shared between it and CPU A. All data sent to the special purpose computer is first loaded into one of the two shared memory modules.

Frame I receives the aircraft's instantaneous position and attitude from the flight computer. Frame I also does some preliminary processing, such as slant range to model centroid tests, to determine the correct level of detail at which to display the models. A model is defined as any collection of object faces for which relative priorities have been determined and stored during off-line data base development. Frame I also performs some data base culling. Objects completely outside the FOV are eliminated. Since geometric and video processing must be performed differently for each display channel, objects are placed into groups according to the display channel for which they may be visible.

The process is called channel assignment. It is accomplished by enclosing each object by a sphere and determining which spheres fall in whole or in part into the pyramidal view cones of the individual channels. The channel assignment operation is achieved with the help of the vector processing hardware of Frame II.

*-Recently upgraded and now known as Advanced Simulator for Pilot Training (ASPT).

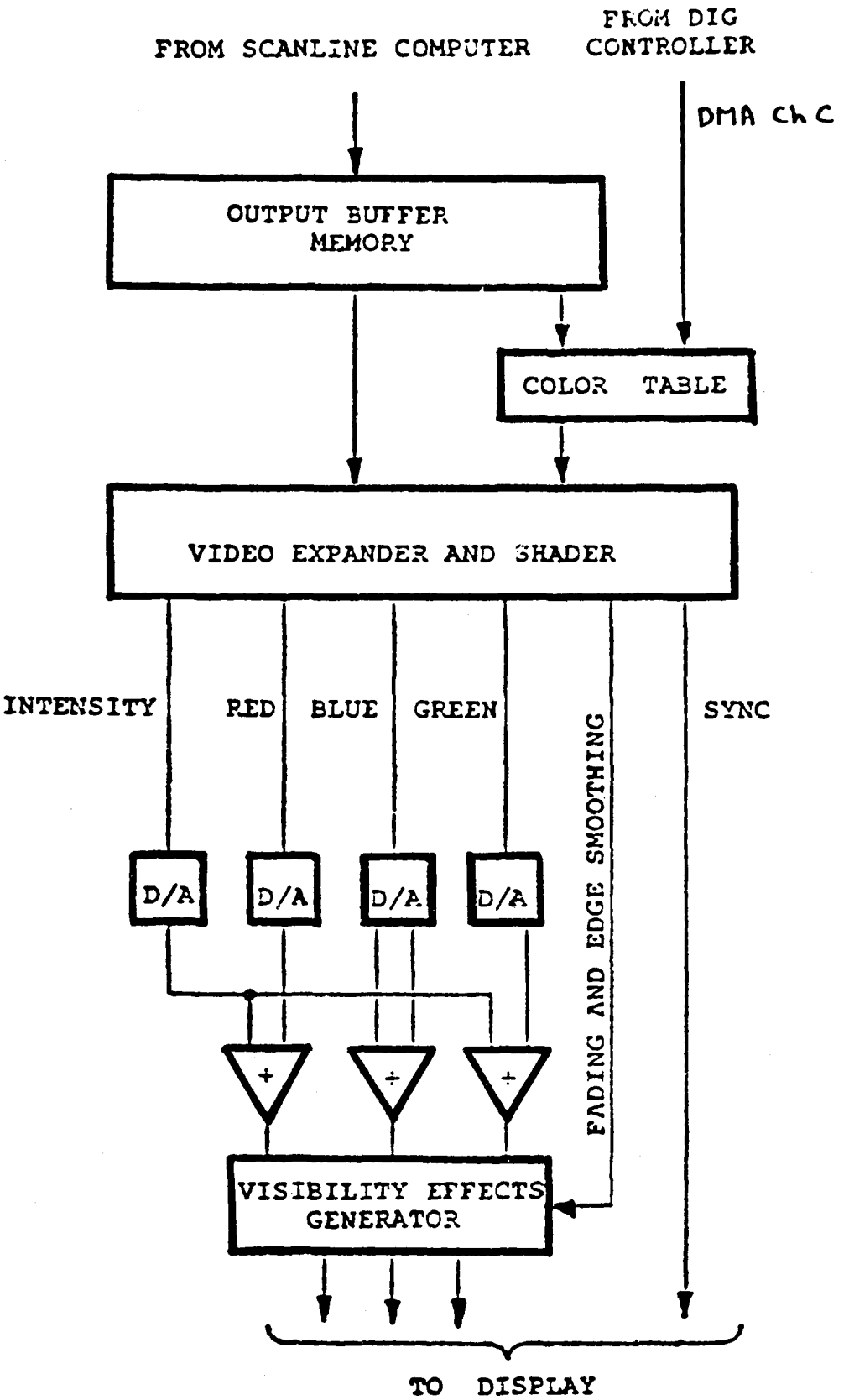


Figure 2.2.1.2.1-7. Output Buffer and Video Generator

The data calculated in Frame I is sent to Frame II via a bi-directional Frame I/Frame II interface. The on-line environmental data storage memories of Frame II are updated by CPU B from data temporarily kept in its memory module.

Frame II is principally a vector processing computer. Every vertex of every potentially visible object is either specified in the environment's fixed coordinate system or in a moving coordinate system. The viewpoint is assigned a moving coordinate system, since it moves through the environment. The vertices and edges in a scene are projected onto each view window, each frame time.

Each moving model, such as an aircraft flying through the environment, is assigned its own moving coordinate system. A set of vectors, joining the viewpoint and vertices of each moving model, is computed each frame time.

The Frame II Edge Processor (pipeline) converts channel specific viewpoint data into a stream of object and channel specific edges. The pipeline processes the edges of each object in sequence. Leaving the pipeline are data describing the projections of edges onto the view windows. When Gouraud continuous shading is used, some special face oriented processing of objects takes place. This is accomplished by passing the object through the pipeline twice, with continuous shading data computed in the first pass and normal edge processing occuring in the second pass.

Hidden surface elimination is accomplished using the priority list approach. A Priority Processor constructs a list containing a unique priority number for each face, based upon viewpoint, active data lists, and separation plane and relative priority data obtained from the data base. The stages of operation are as follows. Viewpoint and moving model positions are received at the start of each raster period from Frame I. Based on this data, the Priority Processor establishes the relative priorities of the coordinate systems. This results in a top-level priority list containing a single entry for each coordinate system. Then for each entry in the top-level priority list, the Priority-Processor constructs a priority list with a single entry for each model of each coordinate system. A complete model priority list is created by combining the individual coordinate system model priority lists, using the relative order of coordinate systems priorities. Next, the Priority Processor creates a separate priority list for each active model. These lists have a single entry for each object of a model. In the final stage, the individual object priority lists are combined in model priority list order to obtain a complete active object priority list. The priority numbers of ground plane faces are then combined with the priority numbers of object faces to form the final list. Since the lower four bits of the priority number are reserved for ground faces and the upper four bits are reserved for object faces, objects always have priority over the planar ground surface.

The Frame II/Frame III interface accepts edge words from the Frame II Edge Processor in arbitrary sequence, provides temporary storage for the edge words, determines the proper sequence of edge words within an object, and transfers this data to the Frame III edge buffer storage.

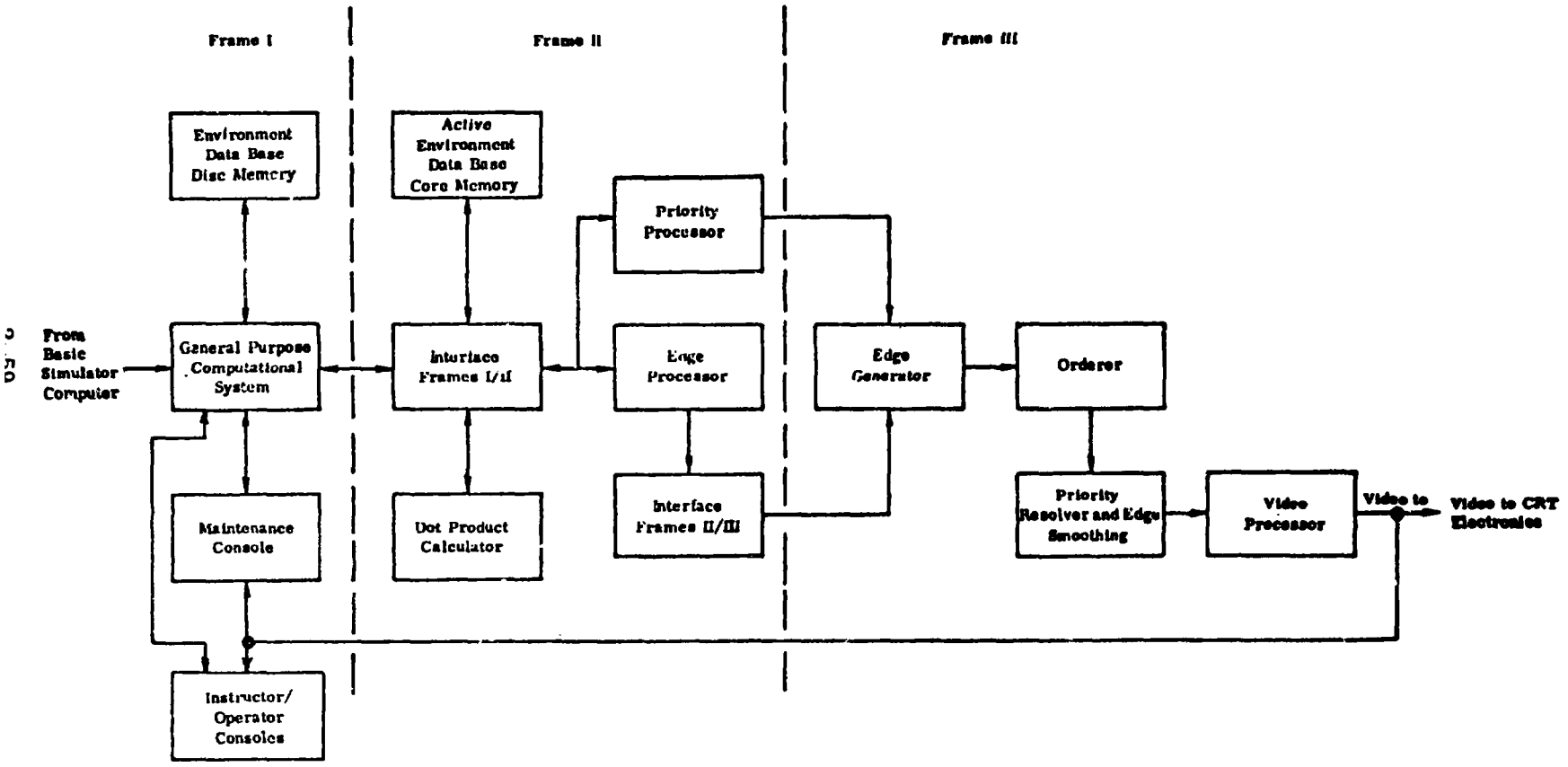


Figure 2.2.1.2.2-1. ASUPT CIG System Functional Diagram

An Edge Generator computes the intersection of each edge with each scan line that crosses it. A maximum of 256 edges per scan line per channel can be handled (a maximum of 2560 potentially visible edges per channel). First, the J-value of the intersection of an edge with the uppermost scan line crossing is computed. The slope of the edge is specified in terms of elements per scan line. The intersection of the edge with each succeeding scan line is accomplished simply by adding the slope to the previous value. The details of implementation are complicated by the 2:1 interlaced scanout used; this complication is ignored here for clarity. The output of the Edge Generator is a sequence of J-values along with their edge identifications.

The Edge Orderer receives edge crossing values for each scan line. These J-values are not ordered across a scan line, except that the values resulting from the same object are in order. The Edge Orderer sorts the edge crossing values by scan element number. This ordered edge data is placed into an Ordered Data Memory.

Final hidden surface calculations are performed by a Priority Resolver. It receives priority lists from Frame II and edge data from the Ordered Data Memory. Priority resolution is essentially a convex object oriented function. When the left element of an object is received for a channel, the object identification is entered in a list. When the right edge of an element is received, the object is removed. Thus, at any moment, the priority list contains the identities of all objects pierced by a ray from the viewpoint, through the raster element being computed, into the 3-D environment. The Priority Resolver uses the available priority information to determine which of these pierced faces is closest to the viewpoint. This determination holds for succeeding raster elements, until there is a change in the list. The output of the Priority Resolver is divided among the display channels, and sent to the Frame III Video Processors for each display channel.

A Video Processor receives one line of data at a time from the Priority Resolver. Edge smoothing is performed in Frame III partially by the Video Processor and partially by the Priority Resolver. The Priority Resolver calculates the position of an edge within the raster element. The Video Processor uses an area times gray level algorithm to determine an intensity value for the pixel. At most, only the values of the two faces of highest priority to the left and right of an edge plus a background value are combined.

ASUPT has 14 Video Processors divided among two cockpits, one for each of the seven channels per cockpit.

2.2.1.2.3 Gould GVS-1 System

Gould has developed a medium performance CIG device capable of a 30 fps update rate (Figure 2.2.1.2.2.3-1). At the front end of their system is a general purpose computer. It performs such tasks as data base retrieval, data management, bookkeeping, communication with the aircraft simulator, and control of the other image generation subsystems (C.3).

The second stage of the system is a Geometric Processor, which performs the 3-D transformations, clipping, and priority processing needed to convert the data base into a form required by the Display Generator.

The Display Generator takes digital inputs from the general purpose computer and Geometric Processor and converts them to full color analog video. The heart of the display generator is a double buffer memory to and from which simultaneous read and write operations are performed. As data from the Geometric Processor is being stored into one buffer, the past frame's data from the Geometric Processor is being stored into one buffer, the past frame's data is being read out of the other buffer. Each buffer consists of multiple memory planes. Each memory plane holds a screen size bit map of image information (but not the image itself). The storage operation into one buffer is performed in random fashion, while the other buffer is being read in raster format. At the completion of a frame time, the roles of the buffers switch. Gould claims that their approach has several advantages over that of GE and Singer (which compute the image "on the fly" in synchronization with the raster scan). Gould's approach reduces sorting and intersection operations, simplifies overload management, and does not place a limit on edge crossings per scan line.

The buffer storage operation is divided into a number of subtasks: face edge outline generation, anti-streaking, face edge priority storage, and computation of edge smoothing coefficients. Face edge outline generation is performed by a Vector Generator using a data list received from the Geometric Processor. This list contains face edge endpoints expressed to subpixel precision. The Vector Generator uses this precision to compute and store edge smoothing coefficients. The Vector Generator extends an edge one full pixel at a time between each pair of face edge endpoints. At each pixel boundary, a comparison is performed between the previous stored data and new edge information. Based on this comparison, those edges or portions of edges that would lead to streaking are removed. The particular memory plane to which a face boundary is written corresponds to the relative occulting priority of the face. A separate set of memory planes stores information which is used to ensure correct definition of face boundaries.

During the read cycle, refresh memory is addressed in raster order. The read cycle performs subtasks of face color filling, hidden surface elimination, and edge smoothing. Face color filling is the procedure which fills the area bounded by the face edge outline, one raster line at a time.

For pixels interior to a face, the fill data for the highest priority face is used as an index to a video look-up table. For pixels crossed by edges, additional logical tests are performed using the stored color and edge information. For this case, simultaneous table look-ups are performed and the data is mixed according to the computed smoothing factors. The output in either case is a digital color code representing red, green, and blue video levels which are converted to analog video by D/A converters.

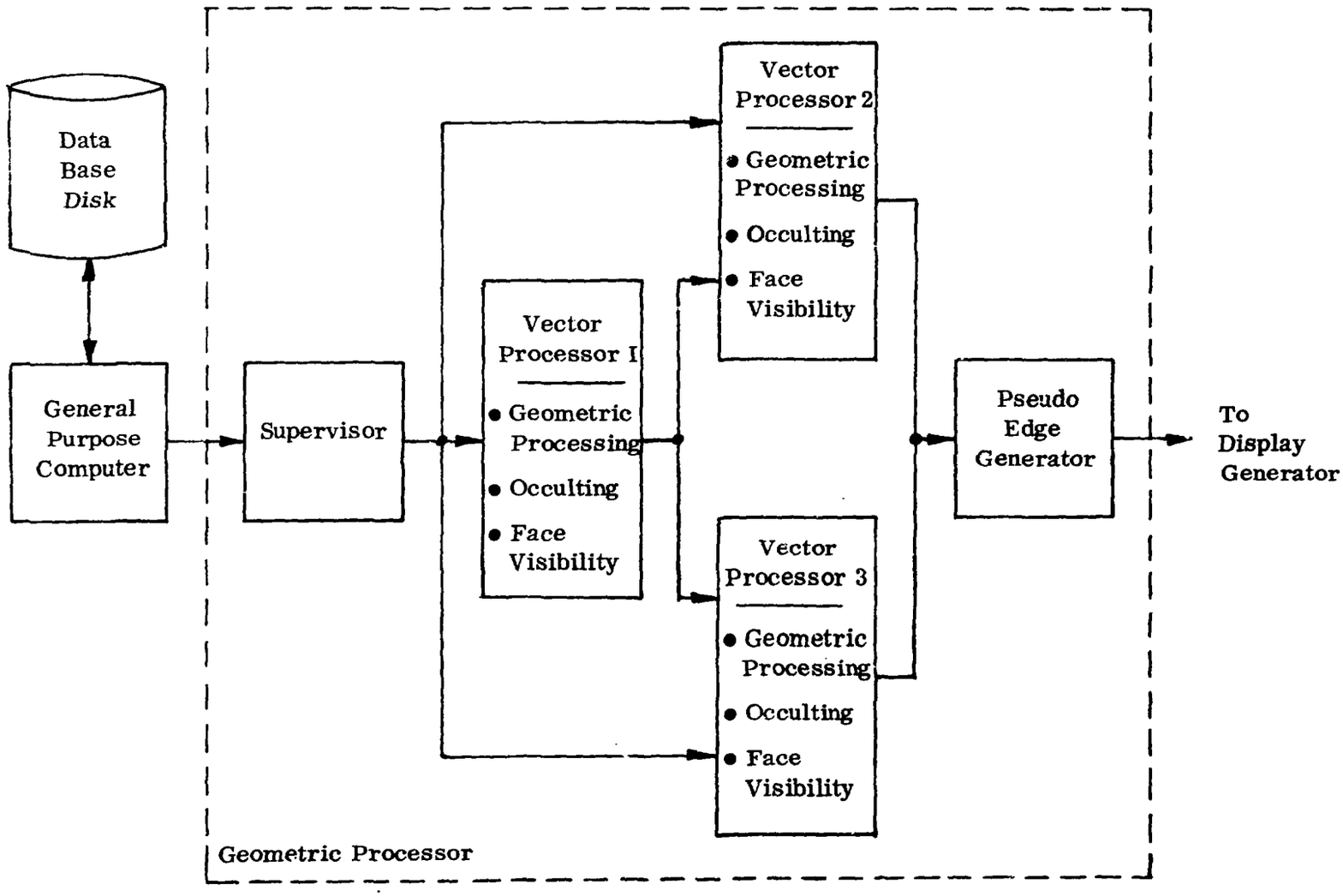


Figure 2.2.1.2.3-1a. Gould GVS-1 System (Part 1)

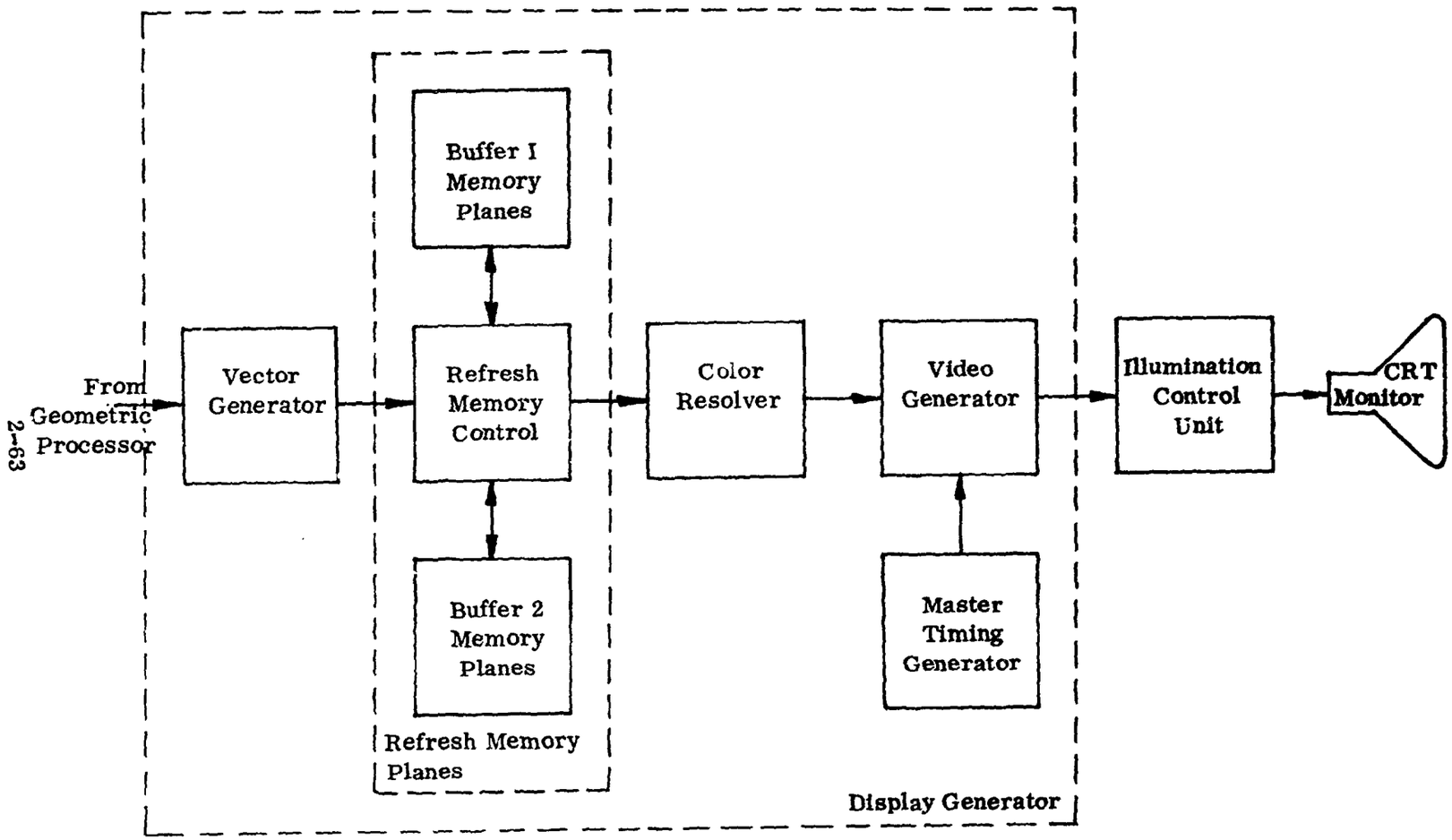


Figure 2.2.1.2.3-1b. Gould GVS-1 System (Part 2)

An Illumination Control unit combines the analog inputs received from the Display Generator, with control parameters obtained from the general purpose computer, to add the atmospheric and illumination effects of horizon glow, haze, and curved surface shading. The output of each Illumination Control unit drives a CRT.

2.2.1.2.4 McDonnell Douglas VITAL IV

The VITAL IV system generates multicolored day, night, and dusk scenes at a 30fps update rate (C.16, C.32). Both the borders and interiors of surfaces are painted in a calligraphic mode, from a palette of approximately 10 colors. Up to 300 surfaces and 8000 light points may be displayed at once. An optical resolution of under 3 arc minutes is obtained over 4096-by-4096 grid. The standard VITAL IV display is a virtual image reflection unit providing an infinity image over a 45^{o} horizontal by 35^{o} vertical FOV. A wider FOV may be obtained by combining up to six displays. VITAL IV's relatively simple design and low cost allows it to be used as a plug-in module for simulator facilities built by other companies. It is often retrofitted into older simulators having no visual capability.

The VITAL IV image generator fits into a single cabinet. It consists of a general purpose computer feeding into a special purpose parallel processor called a Picture Controller (Figure 2.2.1.2.4-1). The general purpose computer takes care of all geometric transforms, hidden surface calculations, and windowing. The Picture Controller consists of a number of special purpose computers and controllers running in parallel. Its operation will be described in more detail below.

Once three-dimensional objects are projected into the viewplane, they are treated as polygons. A polygon is defined in terms of its left and right edge vector outline. A polygon is scanned out by moving the electron beam back and forth between its borders (Figure 2.2.1.2.4-2). Each time the beam reaches a left or right border point, it is displaced vertically to paint out the next raster line segment. Horizontal beam velocity and vertical separation between raster segments are carefully controlled to give a displayed surface a uniform appearance.

A conventional CRT scans out raster lines only n one direction. VITAL's ability to write a line out in both directions allows for a re efficient use of the beam deflection system. Beam position is critical when writi the retrace mode. The very high beam deflection rates used make the brightnes a scanned surface highly dependent on beam dwell time. Horizontal deflection lin ity is critical to picture quality. Variations in horizontal writing rate produce effective changes in beam dwell time, creating intensity variations. When the beam is moved from one raster line to the next, any vertical overshoot may cause the beam to overwrite a previously scanned area. Vertical undershoot reduces the uniformity of the scanned surface. If partial overwriting occurs along the left and right border of a scanned poloygon, the border of the polygon will have an increased brightness. VITAL IV maintains unformity along the border by deflecting the beam a small distance past the border, while blanking out the beam during these excursions. The excursion time outside the polygon is used to

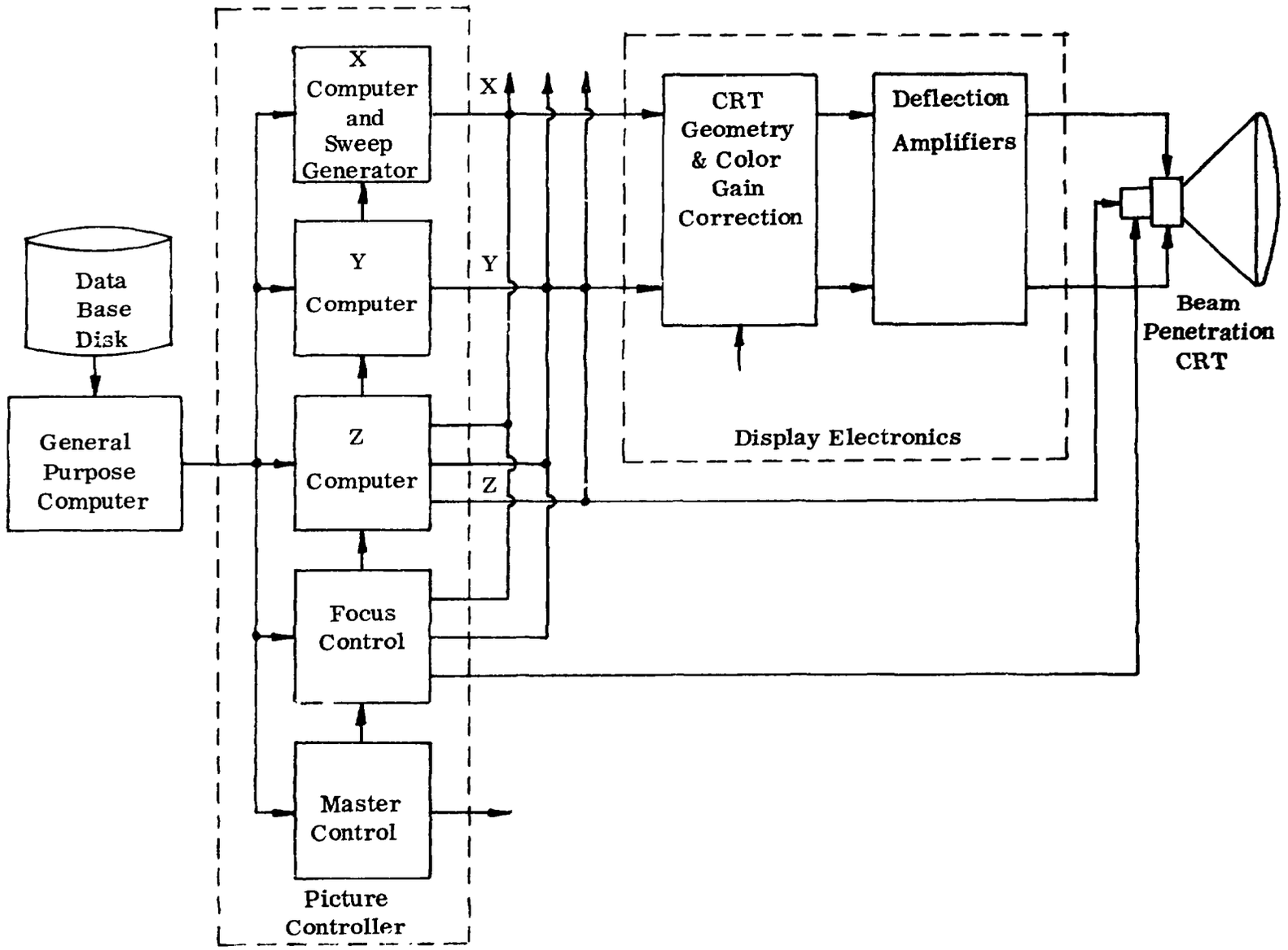


Figure 2.2.1.2.4-1. McDonnell Douglas VITAL Calligraphic Image Generator

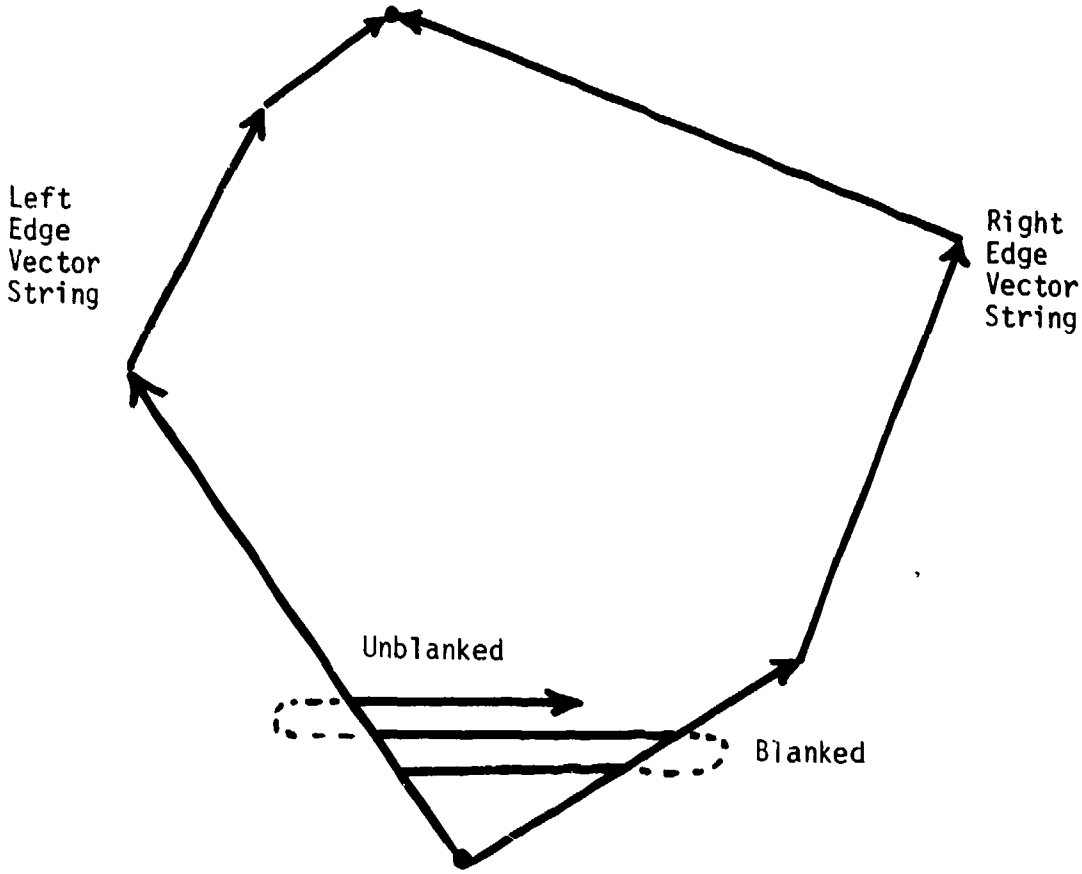


Figure 2.2.1.2.4-2. McDonnell Douglas Polygon Scanning Technique

let the beam settle down before going on to the next scan line. Relative timing is of course critical. The operations of the special purpose computers and controllers which manage these tasks follow.

Image surface generation is handled by X and Y computers and a Sweep Generator. The X-computer calculates the left and right border points for each horizontally scanned surface segment. Each time the beam is moved vertically to paint out the next raster line segment, the X-computer adds a Δx to the previous x border value. Since the borders of polygons are constructed from vectors, each with a slope $\Delta y / \Delta x$, the computation of Δx is a matter of simple mathematics.

The Y-computer keeps track of the current y-value of beam position. It compares the current y-position to that of the y-value of the current border vector's endpoints. When the endpoint of a vector is reached, the proper Δx must be obtained for the succeeding border vector. When the scanning of the polygon is complete, the beam blanks out and jumps to the next polygon to be scanned out.

The Sweep Generator controls the instantaneous horizontal position of the beam.

The Z-computer receives shading coefficients from the general purpose computer. It uses these coefficients to perform a linear interpolation of the intensity values within a surface, as a function of x and y-position. One use of this computer is to provide a tapered horizon glow, fading smoothly as the angle above the horizon increases. The illumination due to landing lights is simulated by tapering the brightness of the runway surface as a function of distance from the landing aircraft.

The Focus Controller manages the focus amplifiers on the CRT's. One function is to display light points of a specified size, regardless of the colors of the lights. Beam defocus is needed to eliminate the visibility of individual lines when filled polygonal surfaces are scanned out. Defocus is used very carefully to prevent edges from losing their sharpness. Special care is taken when scanning out small polygons to keep the beam focused well enough to prevent a fuzzy appearance. For large surfaces, the combination of a lower line density and defocused beam increases scanning efficiency.

2.2.1.2.5 General Electric Second Generation Systems

The General Electric Company was funded by NASA in 1962 to develop a flight simulator for the Apollo Program. A system was delivered in 1963 which displays a simple texture over a ground plane (C.9, General Electric). An improved version built in the mid 1960's was called the Visual Three-View Space Flight Simulator (Figure 2.2.1.2.5-1). It displays three perspective views on three circular RCA color shadow mask CRT's. The three views appear in three non-abutting spacecraft portholes, two with a 60° circular FOV, and the third with a 25° circular FOV.

For the perspective calculations, the vehicle is assumed to have no roll with respect to the raster. The geometric calculations involve projecting the current raster line

onto the ground plane. This projection is equivalent to projecting the endpoints of the raster line. The line segment joining these endpoints is partitioned into 250 smaller segments of equal length.

Binary square mosaic textures are stored by plug-in hard wired diode matrix boards (called map tables). The video is generated by scanning the map tables in synchronism with the raster scan. For video pulse period ($\approx$.2 usec), the position in a map table is advanced an amount corresponding to the length of one small segment. The color value computed for the current map index is then assigned to the scan pixel. This computation will be described in more detail below.

Each texture T is stored at four component levels of hierarchy, denoted by T_1, T_2, T_3, and T_4, where T_1 is the finest texture component. The four are combined by modulo two addition when generating the pattern appearing on the ground plane near the viewpoint. Each successively coarser component is faded out in a zone determined by slant range from the viewpoint. The finest component is faded in zone 1, followed by the next finest in zone 2, and so on until toward the horizon only the coarsest component remains. The coarsest component fades toward a background color at the horizon. No attempt is made to smooth the edges of the colored texture squares.

Fading is implemented by employing two variable gain amplifiers, called A and B (Figure 2.2.1.2.5-2). The inputs to amplifier A are a video signal S_A and a fading value F. The inputs to amplifier B are a signal S_B and a value (1-F). Video signals S_A and S_B are functions of the modulo two sums of the four component texture values. Let

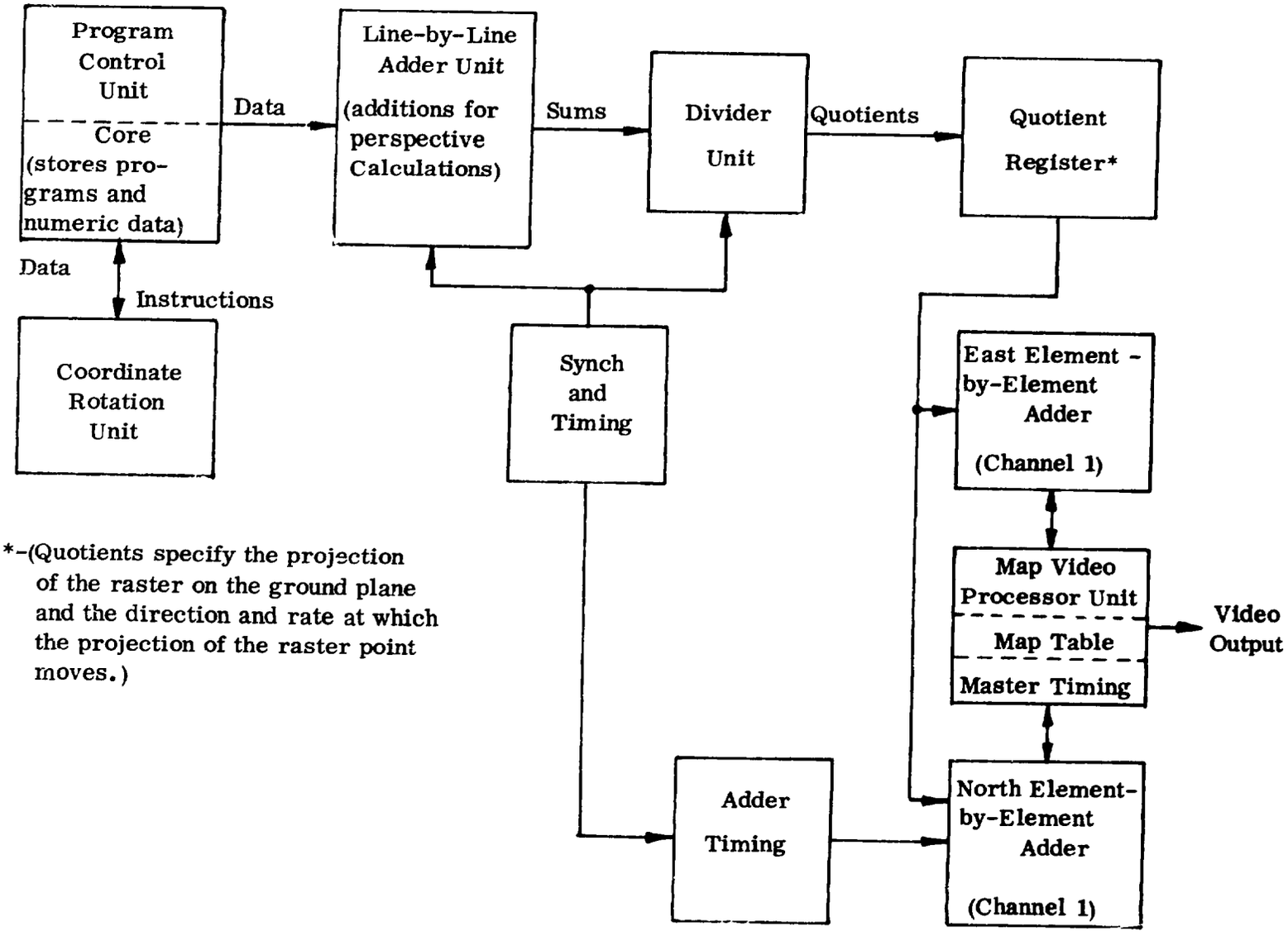
$$S_1 = T_1 \oplus T_2 \oplus T_3 \oplus T_4$$

$$S_2 = T_2 \oplus T_3 \oplus T_4$$

$$S_3 = T_3 \oplus T_4$$

$$S_4 = T_4.$$

In zone 1, $S_A = S1$ is applied to amplifier A with a gain of F and $S_B = S_2$ is applied to amplifier B with a gain of (1-F). As F decreases, amplifier A's gain decreases and amplifier B's gain increases. The result is a video signal in which the contribution of T_1 gradually decreases with slant range. In zone 2, $S_A = S_3$ is applied to amplifier A and $S_B = S_2$ is applied to amplifier B. The T_2 component is now faded as F is increased. The operations performed in each of the six slant range zones are tabulated in Figure 2.2.1.2.5-3.



*-(Quotients specify the projection of the raster on the ground plane and the direction and rate at which the projection of the raster point moves.)

Figure 2.2.1.2.5-1. NASA Three-View Space Flight Simulator

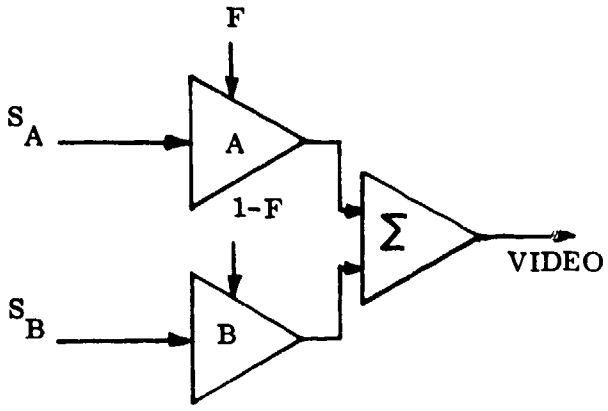


Figure 2.2.1.2.5-2. Variable Gain Amplifiers used for Texture Component Fading

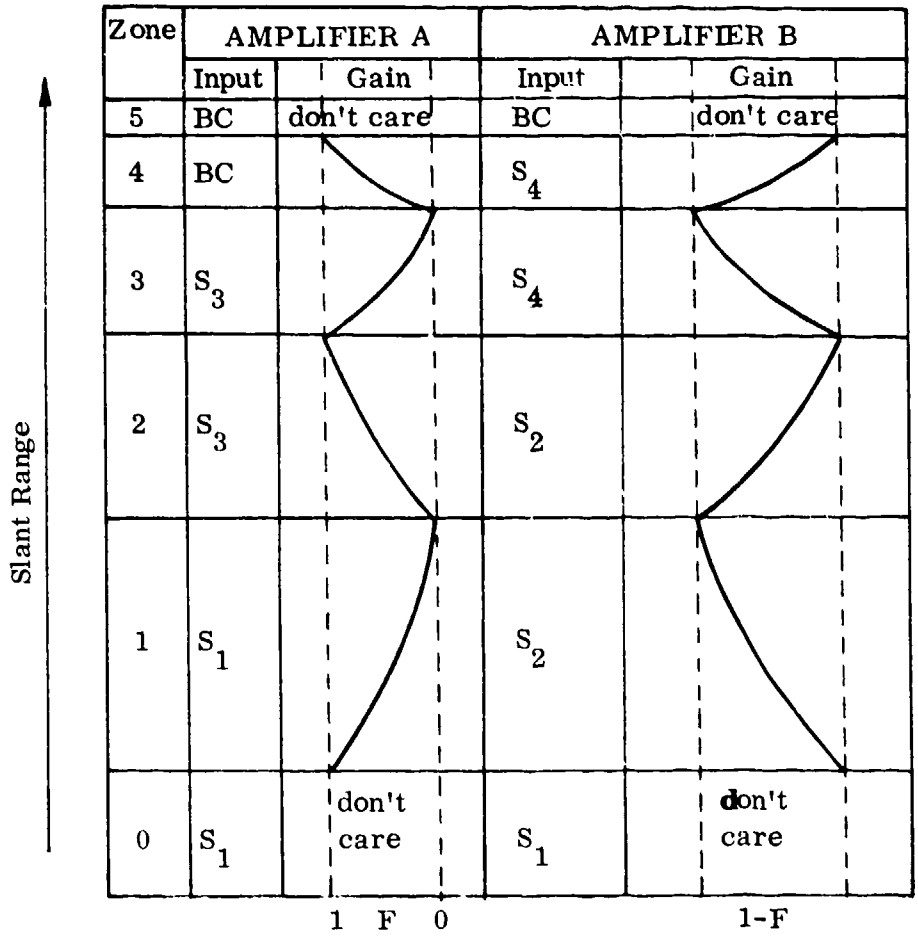


Figure 2.2.1.2.5-3. Summary of Inputs to Amplifiers A and B

(BC denotes background color intensity at horizon)

The night sky pattern of stars is also treated as a texture. A sky texture is placed on a plane at a fixed height above the ground. To make the pattern appear to be at near infinity, the night sky texture does not respond to the simulated vehicle's translation.

The raster structure is identical to that of normal commercial TV, utilizing 525 lines with a 2:1 interlace and 30fps update rate. However, instead of supplying normal horizontal and vertical sweep to the deflection yoke, a composite waveform is applied which is a function of the vehicle's roll.

Later versions of this simulator had the ability to display a limited number of objects. A 1966 report (C.10) describes a unit which can display two rectilinearly oriented parallelepipeds standing on the level ground. Back facing faces of the objects are eliminated from consideration as a result of computing the dot product of the current viewray and surface normals. If the viewray passes through the remaining portion of an object, it must pass through one or more of the top, bottom, or front facing faces. These conditions are denoted by the logical variables T, B, and F. $\overline{T} \cap \overline{B} \cap \overline{F}$ describes the condition when a viewray does not intersect the object.

Let the vertices of a front facing face of an object be denoted by A, B, C, and D (Figure 2.2.1.2.5-4). A viewray starts at the viewpoint 0, passes through the current raster scan point P, and on into the 3D environment. In order to determine if a viewray passes through face ABCD, it is determined if the viewray passes simultaneously above line AB, left of BC, below CD, and right of DA. These four conditions are specified by the logical variables F_1, F_2, F_3, and F_4, respectively.

$$F_1 = \text{TRUE iff } \vec{OP} \cdot \vec{OA} \times \vec{OB} < 0$$

$$F_2 = \text{TRUE iff } \vec{OP} \cdot \vec{OB} \times \vec{OC} < 0$$

$$F_3 = \text{TRUE iff } \vec{OP} \cdot \vec{OC} \times \vec{OD} < 0$$

$$F_4 = \text{TRUE iff } \vec{OP} \cdot \vec{OD} \times \vec{OA} < 0$$

Thus, the condition that the viewray passes through ABCD is $F = F_1 \cap F_2 \cap F_3 \cap F_4$. Consider line AB. To determine which side of AB the viewray passes on, the sign of $Q = \vec{0P}\ \ 0A \times 0B$ is monitored. The vector 0P can be expanded into a row and column sum by the equation

$$\vec{OP} = \vec{OP}_0 + \Delta P\vec{I} + \delta P\vec{J} \ ,$$

where ΔP and δP are the screen components of P and $\vec{I}$ and $\vec{J}$ are unit vectors in the I and J directions (Figure 2.2.1.2.5-5). $\vec{OP}_o$ is the vector joining the viewpoint and the top left screen point. We can write

$$Q = \vec{OP}_o \cdot \vec{OA} \times \vec{OP} + \Delta P\vec{I} \cdot \vec{OA} \times \vec{OB} + \delta P\vec{J} \cdot \vec{OA} \times \vec{OB}$$

$$= Q_o + \Delta Q + \delta Q \ .$$

A hardware module called an Obstacle Side Generator continually monitors the sign of Q for each of the edges in the data base. Logical operations on the resultant information yields the raster scan conversion of the objects.

Since a viewray may intersect more than one face, hidden surface elimination must still be performed. This is accomplished by determining the relationship between the viewpoint and the faces of the object. For example, the top face has priority over side faces except when the viewpoint is below the top. Both objects have priority over the ground plane. Priority between the two objects is determined by the relationship between the viewpoint and a plane separating the objects.

2.2.1.2.6 Advanced Technology Systems (ATS) Computrol

ATS will provide a visual system for the U.S. Navy's new Weapons Tactics Trainer (WTT). The WTT is a simulator system being built by the Hughes Aircraft Company for training pilots to fly the F-18 aircraft. It is a two cockpit system which will primarily be used to train for dogfighting and ground attack maneuvers such as subsonic missions launched from aircraft carriers. The Navy is scheduled to begin training with the device in October 1982.

ATS claims to have a flexible design (Figure 2.2.1.2.6-1), based upon ECL MSI logic, which allows for the display of anywhere from 8,000 to 40,000 edges and at least 10,000 light points, per channel, at a 30fps update rate (C.30). Other claims are that the system will be able to display such special effects as translucency, curved surface shading, fog, rain, tapered horizon glow, and an unlimited number of moving objects.

Unlike other visual systems, COMPUTROL does not incorporate a general purpose computer into its design. A custom-built CPU unit manages the image generation process. The CPU controls arithmetic operations, initiates data transfers, and manages other special control units distributed throughout the system architecture. The CPU is able to handle 25×10^6 instructions per second via pipeline decoding of instructions from an internal demand instruction cache. The CPU unit makes use of other floating point units residing in the system when these are not performing their primary assigned tasks.

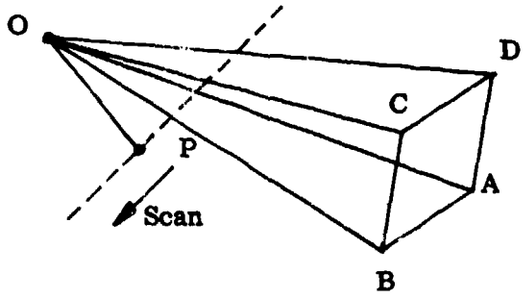


Figure 2.2.1.2.5-4. Viewpoint and Face Vertices Relationship

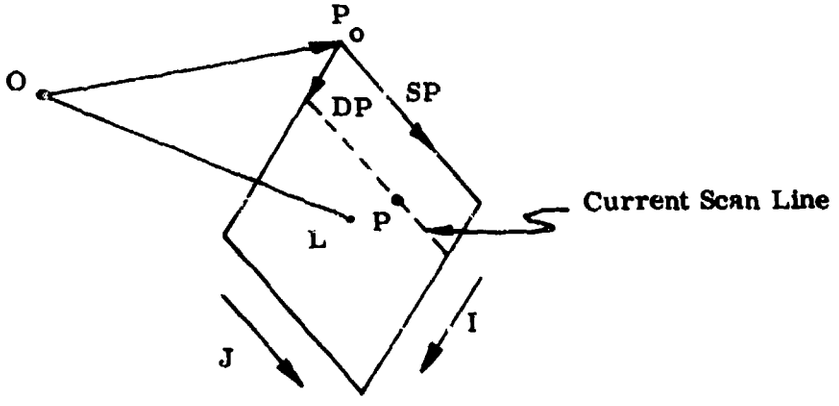


Figure 2.2.1.2.5-5. Viewpoint and Raster Scan Point Relationship

At the heart of the image generation hardware lie two parallel processors: a Projection Processor and a Visible Surface Processor. Each processor consists of very high speed controllers and arithmetic units which communicate with the CPU and main memory.

The Projection Processor retrieves data from disk, projects the 3D environment description onto the 2D view plane, clips edges falling outside the view pyramid, and reformats, sorts, and stores the resulting potentially visible edge and light point data. The Projection Processor consists of a Floating Point Array Processor (FPAP), Edge Assembler and Clipper, and Data Cache.

The FPAP is loaded by the CPU with data and/or pointers to data in cache. Perspective projection is a shared task of the CPU and array processor. The projection of an edge requires a total of 24 multiplies and 24 adds, performed in parallel by two multipliers and two adders. The computation time is 40nsec x 24/2 = 480 nsec per edge.

The Edge Assembler and Clipper first performs clipping by using four comparators to test the left and right or top and bottom of an edge to see if it lies within the view window. If an edge is clipped, approximately 9 floating point additions and 9 floating point multiplies are required to derive new endpoints for the edge. The final tasks are to transform an edge into the form required by the Visible Surface Generator, to construct a system of pointers which will later be used in the ordering of the edges by scanline number, and to place the resulting data into an I-sort Memory (where I is the scanline number). Data specifying an edge includes its starting and stopping endpoints, slope, and shading values.

The I-sort Memory has a ping-pong structure, with each side of the memory operating independently. During one frame time, as data is being written to one side of the memory by the Projection Processor, the previous frame's data is being read from the other side of the memory by the Visible Surface Processor. The I-sort Memory contains over 1.2×10^6 bytes MOS, in which data pertaining to the generated edges are stored and sorted with respect to initial I-value (i.e., the number of the scanline at which the uppermost vertex of the clipped edge resides). Approximately 320 bytes are required to store the data for an edge once it is sorted by I-value.

The Visible Surface Processor takes I-sorted potentially visible edges, and for each scanline, calculates the J-intercepts of the edges. Then it sorts the edge intercepts along a scanline and solves the hidden surface problem on a scanline segment-by-segment basis. These operations are performed by a number of subunits: Fixed Point Arithmetic Processor, Data Cache (visible and current edge buffer), Visible Surface Decoder and Edge Sorter, Visible Edge Encoder, and internal hardwired control logic.

The Visible Surface Decoder and Edge Sorter scans the previous scanline's edge list and adds new edges (i.e., those which start on the current scanline being processed). As each edge is encountered, its J-intercept along the scanline is computed to 14 bits

accuracy, and then sent to a J bucket sort unit which uses 7 bits for sorting, placing the remaining 7 bits of J and a pointer back to the rest of the edge data into cache. The J-sort unit then performs a second pass on the data using the remaining 7 bits. Next, the sorted edge data is again scanned to compute the distance between the viewpoint and each surface segment represented by a scanline segment, to determine which scanline segments are visible. The scanline segments determined to be visible are labeled as such.

The Visible Edge Encoder works in parallel with the Visible Surface Decoder to determine and interpolate color data.

The final stage of processing is handled by a Display Memory and Edge to Raster Decoder Unit. The unit receives visible edge blocks for the 2D scene. It buffers and converts this data into analog video.

The WTT enclosure consists of two 40 foot diameter domes, each surrounding a simulated F-18 cockpit. The inside surfaces of the domes are large spherical screens. Each dome will contain five G.E. light valves (1024 line by 1024 pixel) to project the generated imagery. Three light valves will project a hemispheric sky-earth scene, the other two will provide missile, target, and gunfire imagery.

2.2.1.2.7 Evans and Sutherland Computer Company's Continuous Tone Five System (E&S CT-5)

E&S's CT-5 device is the fifth model in their line of raster scan visual systems (C.26). Two six channel modules are being built for Reflectone, Inc. for use in helicopter trainers.

The first stage of the visual system is a minicomputer which performs such tasks as data base retrieval and overload management (Figure 2.2.1.2.7-1).

The second stage is a Viewpoint Processor which consists of an Object Manager, Polygon Manager, and two independent memories. The Viewpoint Processor performs tasks specific to the momentary viewpoint, rather than to any particular view window. It executes various data base culling operations including FOV inclusion, rejection of back facing faces and faces smaller than the allowed threshold. The Viewpoint Processor feeds a Geometric Processor which performs view window specific computations including rotation, clipping, and perspective divisions. The resultant image plane data for a channel is stored in a dual Polygon Buffer. The buffer contains view window descriptions of polygons and/or lights. Scene components are arranged in priority order. While one side of this dual memory is being written into at a field rate (normally), a complete scene's worth of data (i.e., both fields) is being read from the other side.

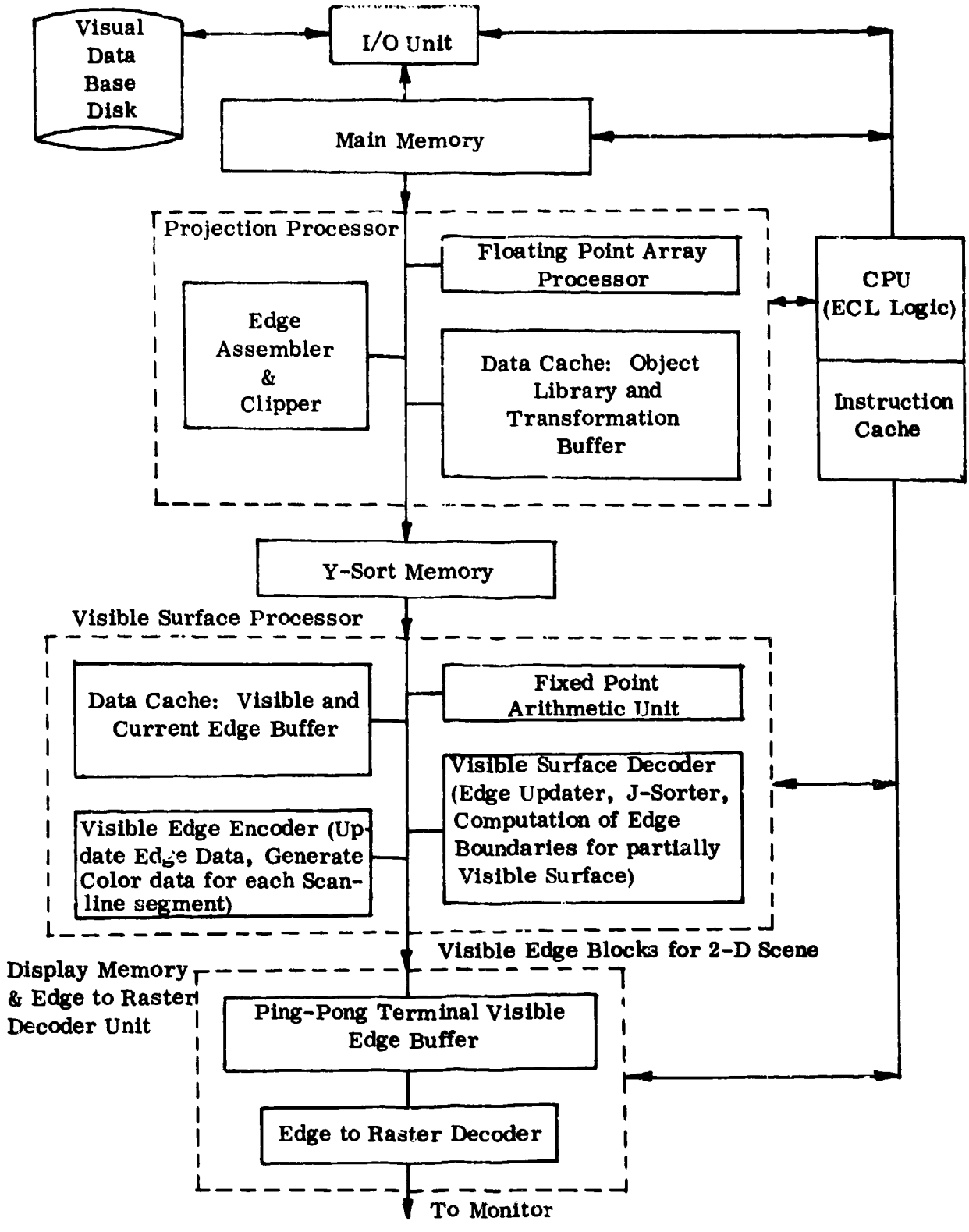


Figure 2.2.1.2.6-1. ATS Computrol System

The Polygon Buffer supplies 2D geometric data to a Display Processor. The Display Processor receives and processes scene components in priority order. Thus, scene components received earlier will occult later ones.

The Display Processor consists of a Span Processor, Mask Memory, Spatial Filter, and Video Assembler. It performs hidden surface elimination, anti-aliasing, and raster scan conversion. The CT-5 system is the first raster scan CIG device built by an established company in the business not to use a scanline approach to scene generation. Images are computed in feature sequential order, rather than scanline order. Area representations are the basic unit for processing rather than scanline samples. The processing hardware is organized into units which operate in parallel on regions of the display called spans. A span is a cell of a rectangular tessellation of the view window (Figure 2.2.1.2.7-2). Each span, in effect, represents an array of pixels. Individual scene components such as polygons are processed in span sized chunks. The algorithms which do the processing for a span operate on analytic descriptions of scene components, rather than sampled versions of them. E&S claims that the time required to process a span is essentially unrelated to the complexity of the geometry within it. As a consequence, the time required to process an object is proportional to the number of spans which it intersects in its view window projection.

A Mask Memory stores a high resolution composite record of all scene components processed thus far for a field. Only information indicating the existence or absence of scene data is stored. When a polygon is submitted to the Span Processor, an analytic description is formed of the portion of the polygon covered by the span. A description of image regions already processed for the span is obtained from Mask Memory. The new polygon, minus the higher priority mask area, is passed to a Spatial Filter. The inclusive OR of the new polygon with the mask becomes the new contents of the Mask Memory.

The Spatial Filter re-combines a high resolution geometric description with its associated shading data. The result is spatially filtered and sampled to yield video data at pixel resolution. E&S has not published details of the anti-aliasing equations implemented by the filter.

The Spatial Filter feeds span size arrays of video, in parallel, to a Video Assembler. The Video Assembler is a dual video buffer with storage for the red, green, and blue components of an image at display resolution. While one side of this buffer is supplying video to a display, the other side is accepting scene data for the next field.

A number of common approaches to overload management are applied in combination. The size threshold for accepting faces is dynamically adjusted to control the number of polygon passing through the pipeline. Level-of-detail selection parameters and the ranges at which switching occurs are also adjusted when necessary. If conditions deteriorate so rapidly that the hardware units cannot complete their tasks in the allocated time, the field rate is lowered. The field rate is not decreased below the frequency for which excessive flicker will occur. Under the most severe conditions,

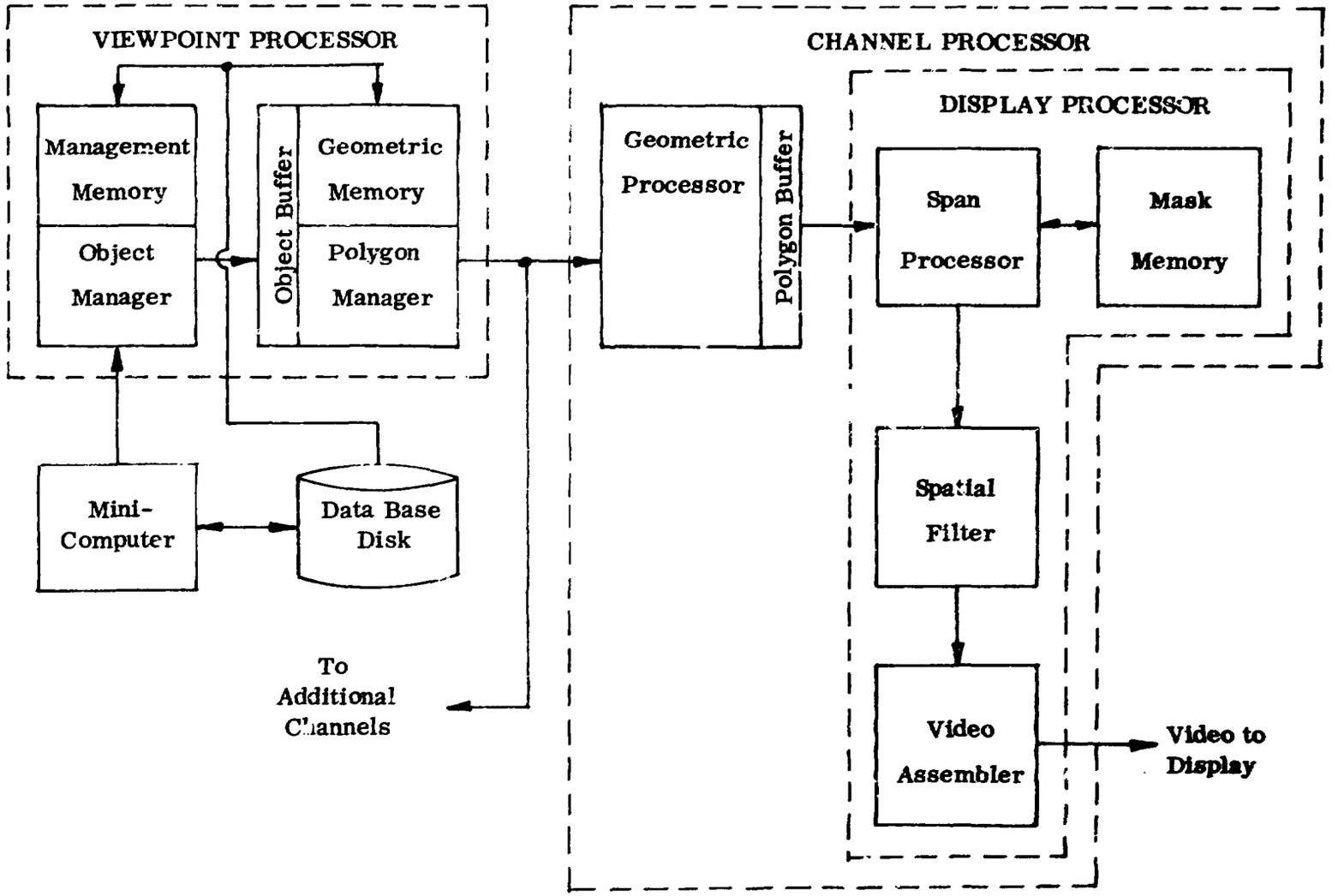


Figure 2.2.1.2.7-1. Evans and Sutheriand CT-5 System

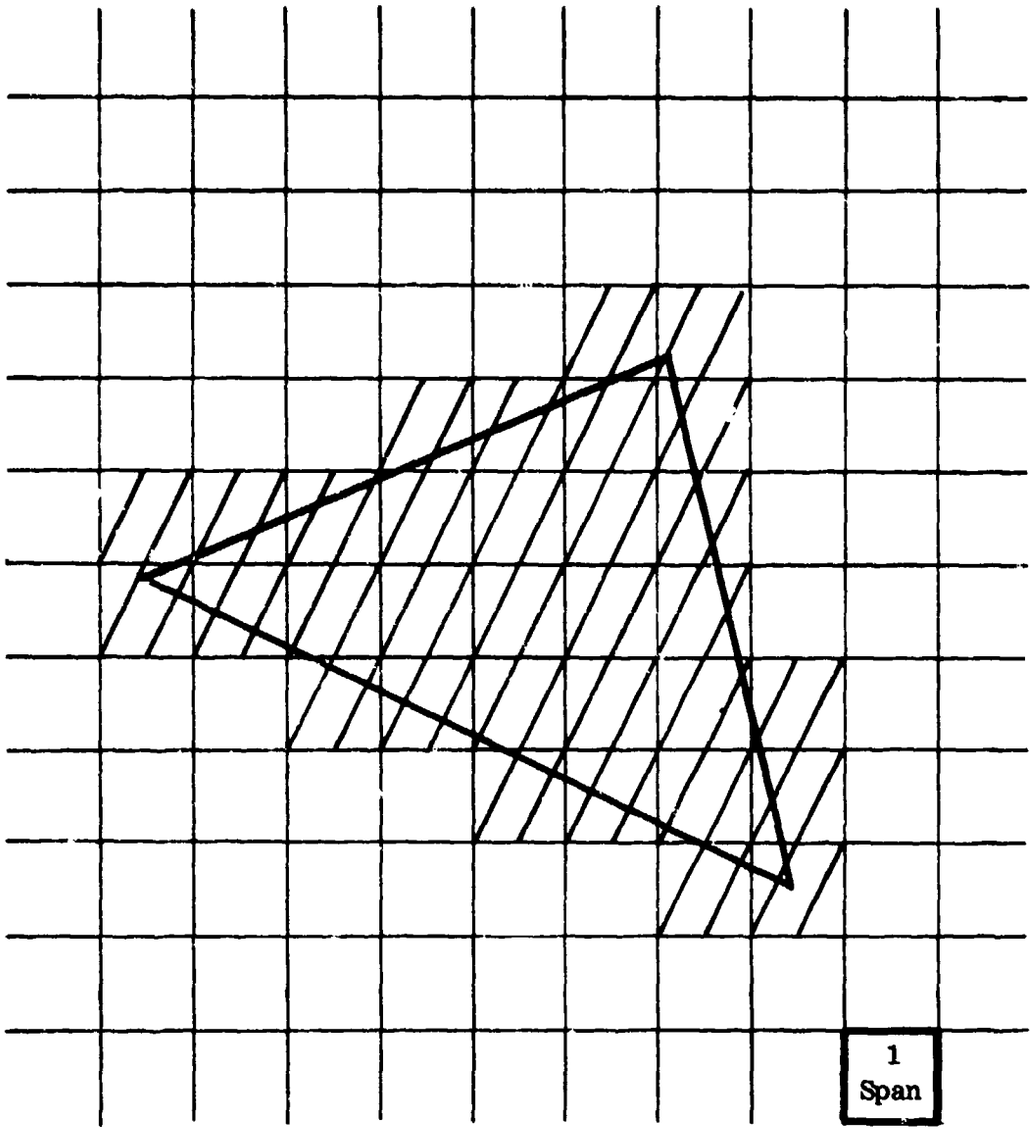


Figure 2.2.1.2.7-2. Dissection of a Polygon into Spans

a frame rate update is employed. This doubles the allocated processing time for all hardware units up to the Display Processor. As a last resort, the Display Processor succumbs to a frame rate update, but still with a field rate display.

2.2.1.2.8 Marconi Radar

Marconi Radar Systems has announced the development of a 50 frame per second computer image generation device which incorporates some unique features (C.20). Data bases are created by hand using a digitizing tablet, with most of the environment being constructed from triangular faces. Definitions are contained in a vertex list, a face list, face normals, face colors, texture data, and a model list.

The hardware approach is shown in Figure 2.2.1.2.8-1.

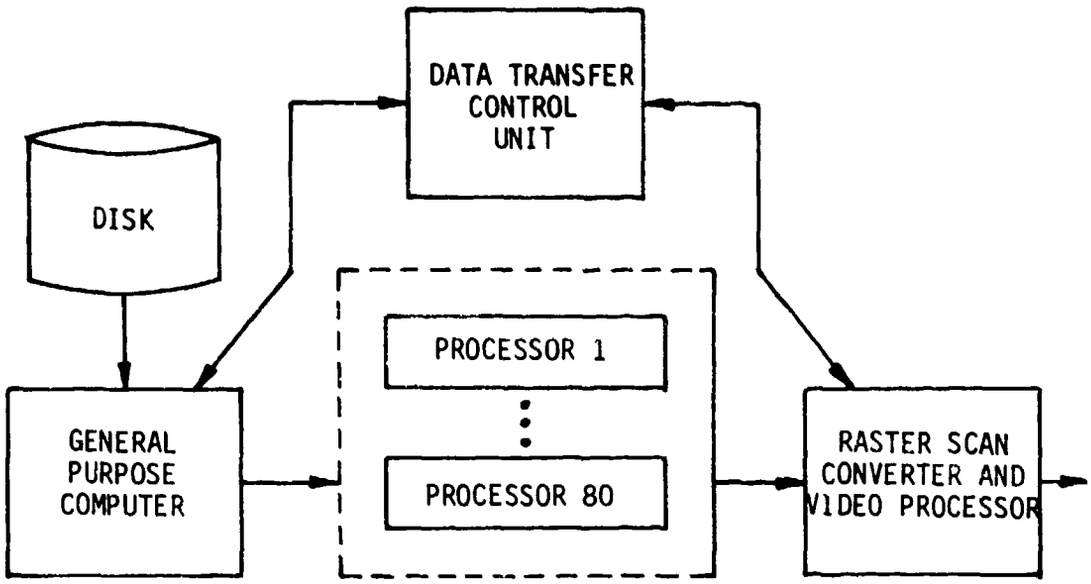


Figure 2.2.1.2.8-1. Marconi Computer Image Generation Approach

Frame I: The general purpose computer manages the flow of data into the system. In particular, it supervises the transfer of data blocks from disk to their on-line environment data store memory, and selects models of the correct level of detail based on range and system loading. System overload can be detected. Less detailed models are selected to provide a graceful degradation of the system under peak load conditions.

The speed of operation of the Frame II processor makes a high speed interface necessary to transfer model data by direct memory access at both source and destination. A data control unit is used for this transfer, and is managed by Frame I.

Frame II: The Frame II hardware consists of up to 80 identical processors, each handling 15 polygons. Each processor is a special-purpose unit with a 16 bit word length and an instruction cycle of 100 nanoseconds. The sequence of operations within each processor is repeated every 20 milliseconds. Hidden surface elimination is accomplished by the priority list approach. A 4 x 4 subpixel approach to anti-aliasing is employed.

Frame III: In Frame III, several dedicated processors are pipelined to provide an increasing data flow rate as data progresses down the pipeline. The first process is repeated every 20 milliseconds. Data is sorted by channel and edge values are computed. This edge data is placed into a field buffer store. The second process takes place at the line rate. It is a computation of the positions of edge crossings with the current scan line. These edge crossing points are called "change points".

The third process also occurs at line rate. It orders the change points, and by a parallel process computes control parameters for edge smoothing and texture generation. During a fourth process priorities are evaluated.

The final set of processes take place at pixel rates. During a pixel period, computations are performed using color, transparency, texture, shadings, and edge smoothing data. Color assignment is made with 21 bits per pixel.

Display: Eight separate displays can be combined, each having a 40^{o} x 30^{o} FOV.

Special Effects:

Real world textures are compressed with a fast Walsh transform (see details below). The system applied the inverse transform to the compressed data. The result is placed onto image surfaces. The texture is displayed at three levels of detail, in correct perspective, with blending between texture levels of detail.

Smoke is implemented with translucent faces. An interesting concept representing forests involves use of a raised textured translucent face to represent the tree tops. The ground is visible through the holes in the texture pattern; moving tanks can pass under this raised surface. Curved surface shading is also implemented.

Texture Generation Approach:

The fast Walsh transform functional basis consists of matrices of +1's and -1's, defined as follows:

$$\text{Matrix of order 2} = W_2 = \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}$$

$$\text{Matrix of order 4} = \left[\begin{array}{c|c} W_2 & -W_2 \\ \hline W_2 & W_2 \end{array}\right] = \begin{bmatrix} 1 & -1 & -1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

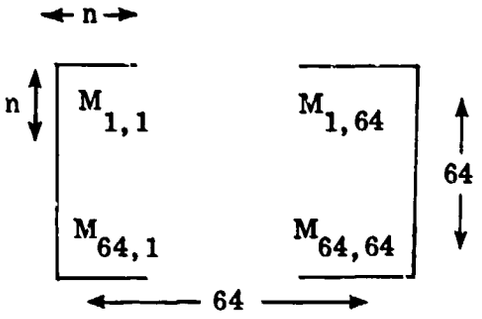
$$\text{Matrix of order 8} = \left[\begin{array}{c|c} W_4 & W_4 \\ \hline W_4 & W_4 \end{array}\right]$$

etc.

Let T represent a 64 x 64 prototype texture. This texture can be an array of intensity values taken from a sampled real world texture (such as an aerial photograph of a forest). The Walsh transform of the texture T is specified by the multiplication of three matrices.

$$\text{Walsh transform of texture array} = M = W_{64} \; T \; W_{64}^{t} \quad (\text{Note: } W^t = W)$$

Here we let M denote the texture in Walsh transform space. M is a 64 x 64 matrix. It can be computed off line and stored. However, the entire matrix need not be stored. All that is needed is a compressed version of M. A compressed version of M simply consists of an nxn sub-array of M, fixed to the upper left corner.



The inverse of a Walsh transform matrix equals the matrix itself. That is, $W^{-1} = W$.

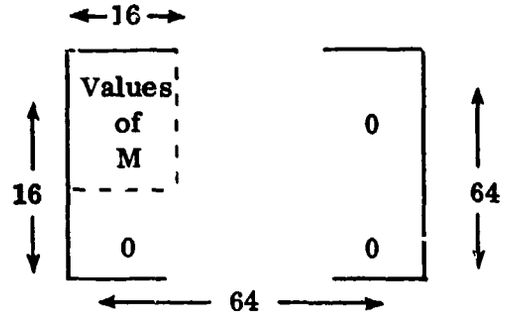
To place the texture on an image surface requires the computation of its inverse Walsh transform: $T = W^{-1} \; M(W^{-1})^{t} = WMW$.

That is, a number of additions and subtractions are applied to the stored texture matrix, since the Walsh matrix is only composed of +1's and -1's.

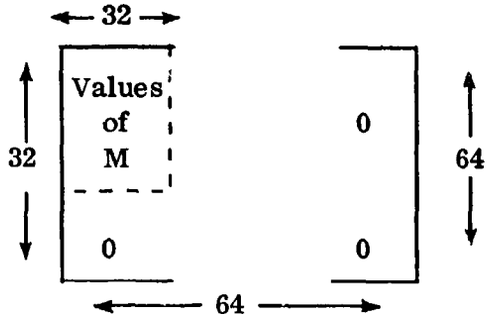
In actuality, the number of operations is reduced considerably by applying the inverse transform to the compressed version of M.

It is very easy to generate different level of detail versions of the texture. This is because the low frequency components in transform space are toward the upper left corner of matrix M. The high frequency components are toward the lower right corner of matrix M. A lower level of detailed version of M simply consists of only using those components of the matrix near the upper left hand corner.

Lower level of detail version of M =



Higher level of detail version of M =



2.2.2 IMAGE PROCESSING AND SENSOR SCENE SIMULATION

The field of image processing includes a bewildering variety of techniques whose common goal is the extraction of meaningful information from extremely rich geometric inputs (V.25 Rosenfeld and Kak, 1976). These inputs are images, characterized as regular mosaics of intensity measures (brightness, temperature, color, etc.) typically consisting of hundreds of thousands of bytes of information each. The desired outputs are usually a few dozen bytes of information necessary to specify the location and type of a few important features such as targets, threats, and landmarks. One of the goals of this study is to apply such drastic compressions of geometric information in representing scenes. Compressions of this magnitude have already occurred in defining

data bases used by sensor simulations, so there is not much room for improvement. That is, the raw data for a sensor data base may be photographic or radar images of terrain. These sources have traditionally been reduced to maps or digital data bases by hand; more recently, automated techniques have been developed to aid in this reduction process. In the map or digital form, digital image information is reduced to elevation and planimetric data in a far more compressed form than the original data. Any further reduction of planimetric data would lower the quality of the representation. Attempts have been made to reduce the data necessary to represent terrain grids by polynomial approximation (D.9). Though a five-to-one reduction in data is claimed for certain types of geographic areas, violence is done to discontinuities in terrain such as cliffs and canyons. These are precisely the kinds of vivid visual features which must be preserved in sensor simulation.

One possible application of image processing is in the data base preparation stage. Edge enhancement and region segmentation can be used to infer significant structures such as roads and extended areas of a particular type of surface material. Most of these techniques have failed to perform in a totally automatic fashion; rather, they are used as aids to human performance of image interpretation. Pattern recognition and image understanding exhibit fundamental difficulties in complex real environments because the performance goals are inherently tied to those of artificial intelligence (V.32 Winston, 1975). Whether the application be language translation, artificial vision, or problem solving, none of the many ingenious methods of artificial intelligence have successfully demonstrated true intelligence in man-made devices applied to real world problems. There are aspects of open-endedness, creativity, and generalization present in natural intelligence which no artificial scheme has yet been able to demonstrate except in the most artificial of environments. This problem and some possible alternate approaches to artificial vision based on transformational geometry are discussed in Reference Material IV.1 (V.31).

There are many valuable techniques in image processing which could be used to enhance sensor imagery. Typically sensor images are of lower resolution and color quality than out-the-window daylight views because of the nature of the sensor (F.2, F.6). Image enhancement techniques such as histogram modification, edge detection, threshholding, and filtering could greatly improve the pilot's view of the world through sensor displays. These techniques could be tested in software in a laboratory sensor simulation system such as that described in Section 4 of this report.

SECTION 3

TECHNIQUES EVALUATION

3.1 POLYGON DATA BASES

3.1.1 SCENE DATA STRUCTURES

3.1.1.1 Introduction

This section describes efficient data base structure for the types of scenes that a pilot/navigator sees or an electro-optical sensor captures during low-level flight. The three main components of any scene of this type are terrain (surface topography), culture (ground material cover), and three-dimensional (3-D) objects. A scene model is a specification of the way in which these components are represented in the computer. The scene data bases fitting this model should be efficient in terms of both storage and processing because they will drive the image generation system.

3.1.1.2 Representation of Terrain and Culture

3.1.1.2.1 Terrain Representation

In the real world, terrain is a continuous surface. Since no computer model can specify the land surface exactly, the usual objective is to obtain a satisfactory approximation, which minimizes the effort required to develop, store, and display a data base.

One useful source of data is the Defense Mapping Agency (DMA) digital terrain elevation files. Each terrain file contains an array of elevation values for a region of the Earth. Since a matrix array stores only the altitude of the surface at each sample point, geographic locations are determined by grid spacing, and are implicit in the sequential positions of the altitude values within the storage array. One advantage of this method of storage is that the neighbors of a sample point are easily located. The principal disadvantage of a grid is the certainty of redundancy, since the grid must be made sufficiently dense to portray the smallest terrain undulation of interest. To improve the resolution of a grid by a factor n, the grid spacing must be decreased by this factor, increasing the number of sample points by a factor n^2. Smooth subareas then contain far more sample points than is needed to portray them. Other problems arise when displaying grid data using conventional ray tracing methods. When a

surface is displayed in perspective, areas close to the viewpoint will have a "blocky" appearance. For regions close to the horizon, a very large number of terrain points will cover but a single picture element (pixel). This can cause aliasing problems and a tremendous increase in the number of computations needed for display. The usual solution is to place a false horizon nearer to the observer than the actual true horizon. Algorithms for direct display of grid data will be described in paragraph 3.2.

Contours represent another way of sampling and storing a terrain surface. The elevations of contours are defined relative to sea level. Contour maps are available for many areas of the Earth. These may be placed onto a digitizing tablet and input to the computer by tracing the contour lines with a digitizing stylus. Problems associated with storage and display of digitized contours have led most researchers to use contours as a data capture technique, converting to other structures for long term storage (D.12).

Another approach to storing terrain is the polynomial patch (D.9). Here, specific mathematical functions are developed to describe each patch of interest. This usually involves the fitting of low order polynomials to small terrain squares. Weighting functions assure continuity at patch boundaries. The polynomial patch is not an appropriate method for terrain representation, however (D.11, D.12). This approach has not arisen from the phenomenon of interest (i.e., the Earth's topography), nor even from the data domain or goals of the analysis but rather from computer graphics techniques. Patches may provide useful estimates of average quantities, but they cannot generally accommodate slope discontinuities or local roughness changes conveniently without being redundant over most parts of the data area.

Terrain points are termed surface specific if their location depends on local surface topography. Features of this type include peaks and pits (local maxima and minima), passes or saddle points, and ridges. Surface specific points require more storage space per point than grid points, since all three coordinates (ground position and elevation) must be explicitly specified. Surface specific points can be obtained by applying digital filters to DMA terrain arrays. Such filters have been designed by General Electric for a number of projects. Surface specific points can also be input to the computer manually, through a map mounted on a digitizing tablet.

Surface-specific points can be joined to form a network of triangular faces. These faces in aggregate form a piece-wise planar approximation to the Earth's relief. This triangular representation is convenient for a graphic processing and display. One shortcoming of triangular faceted surfaces is that they are discontinuous in slope at edges. This produces an unpleasant appearance in display and creates forbidden zones where 3-D objects cannot be placed due to the nature of the fast hidden surface algorithms used.

It is possible to form a triangulation directly from a contour map, but the result is not as efficient as surface specific methods.

3.1.1.2.2 Culture Representation

Planimetric man-made and ecological surface features are often categorized as culture data. Some typical features are lakes, forests, swamps, fields, deserts, and farmland. Terrain data specifies the topography of the Earth, while culture data indicates its color and texture.

The DMA supplies culture files for certain regions of the Earth. A culture file contains mainly the polygonal borders of culture features, along with encoded descriptions.

Culture data can also be obtained by digitizing maps and aerial photographs.

In order to color a triangular terrain surface, each culture polygon residing in the area must be intersected with the triangular network. Each polygon must then be broken up into fragments, such that each fragment falls on only one triangular face.

Another approach to representing the cultural content of a terrain triangle uses a Voronoi mosaic. A small number of Voronoi points are placed onto each terrain triangle (Figure 3.1.1.2.2-1). Each tile of the mosaic consists of the geometric region whose elements are closer to the single Voronoi point at its center then to any other Voronoi point. Each of these points represents a different culture type. When the triangular face is displayed in color, each spot on the face is assigned the color of the nearest Voronoi point. Thus, only one point is required to describe an entire culture polygon. This yields an efficient method of storing culture data. Appendix IV.2 provides a detailed description of Voronoi tessellation.

3.1.1.2.3 Terrain and Culture Storage

Approaches to terrain and culture storage were reviewed in paragraphs 3.1.1.2.1 and 3.1.1.2.2. In surface specific sampling, the points which are selected have a particular topographic significance; these include peaks, pits, and passes, and points along ridges and valleys. In order to display the terrain by computer, these points must be connected to form a network of triangles. In the surface random approach, points are selected according to criteria independent of the topography. Usually points are selected on some sort of grid or the elevation of the points are fixed (contour sampling). Generally, grids require much less storage per point than other methods, since only one quantity (elevation) must be stored per point. Digitized

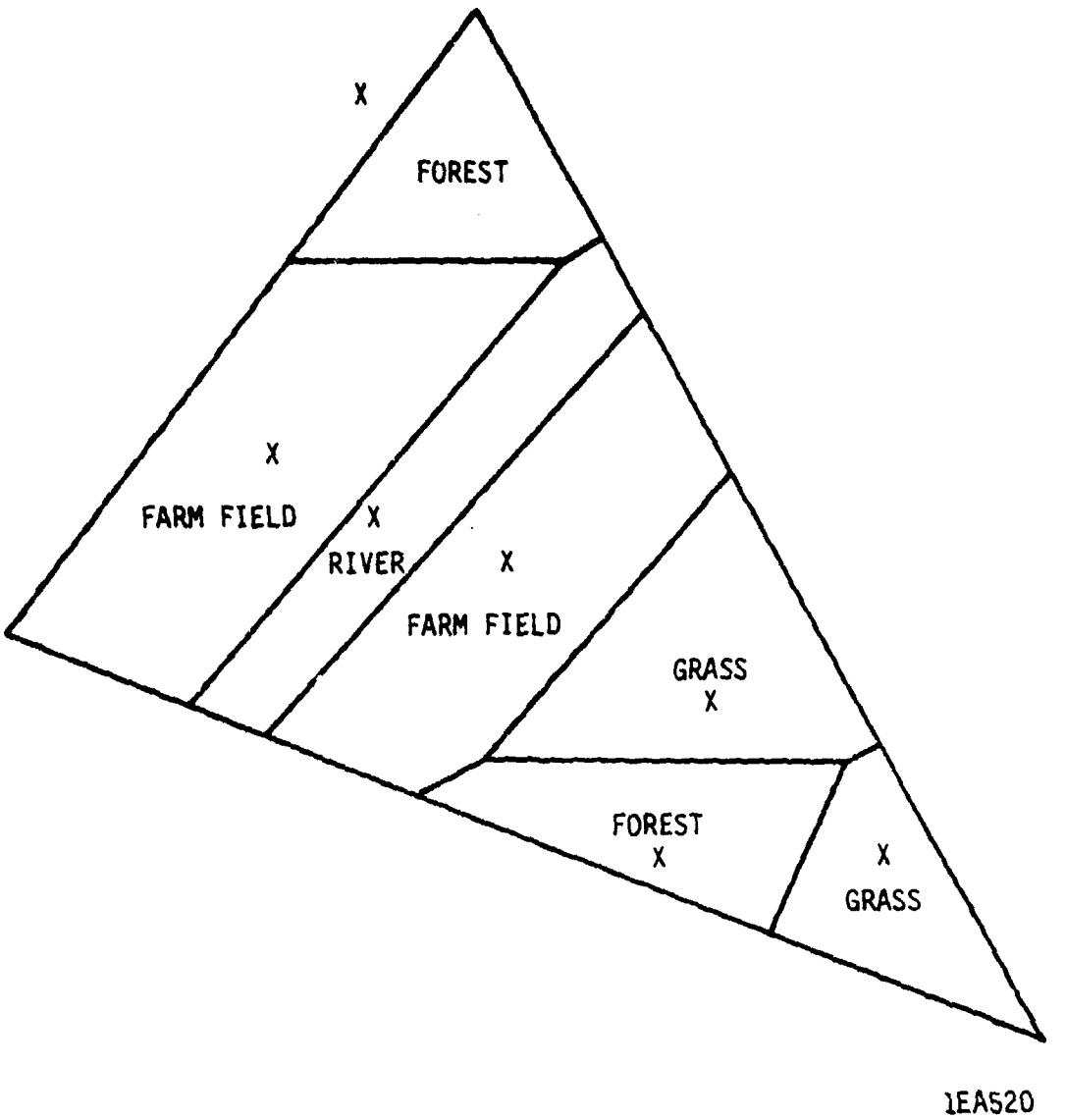


Figure 3.1.1.2.2-1. Cultural Content of Terrain Triangle Represented by Voronoi Mosaic (Cells Centered at X's)

contour points require two coordinates, while all three must be specified for surface specific points.

In the final analysis it is not the amount of storage required per point that is the determining factor in the choice of a terrain storage scheme, it is the total storage required. It can be shown theoretically that in order for grids to produce a level of precision comparable to that of surface specific points, more storage is required (D.11). Furthermore, it is easier to place 3D models on top of planar faces than a continuously changing surface. For these reasons, a triangular terrain model appears to be a reasonable choice for many applications.

Culture polygons may be represented by the strings of vertices defining their borders. Storage can be reduced by eliminating the least significant vertices from polygonal boarders.

3.1.1.3 Representation of Three-Dimensional Objects

Three dimensional objects should consist of convex polyhedra because convexity assures simple hidden face calculation. To completely specify any polyhedron requires both a topology of the adjacency relations between the polyhedron's constituent parts and a geometry specifying the physical dimensions and locations in space of each part. The geometric description of a polyhedron involves data, while the topological description involves structure. Without both types of descriptions an object's shape cannot be completely represented, and some operations are impossible.

The topological and geometric aspects of a scene model must be represented in the computer in a form that allows convenient access by the algorithms that generate images. The precise choice of a data structure cannot be made until a viewing algorithm is selected. For example, some hidden surface algorithms need to know which surfaces meet at a particular edge, while others do not require such information. Similarly, some make use of groupings of faces into objects or clusters, while others treat faces independently.

3.1.1.3.1 Geometrical Representations

Classical cartesian coordinate geometry provides a number of schemes for representing a polyhedron. A vertex can be represented by a point, a face by a plane equation, and an edge by a pair of points (Table 3.1.1.3.1-1).

The geometric definitions of face, edge, and vertex can be derived directly from each other (Figure 3.1.1.3.1-2). A vertex is the intersection of two (coplanar) edges. An edge is the intersection of two faces or joins two vertices. A face can be computed from two (coplanar) lines or three vertices. Thus, many combinations of vertex, edge, and face geometrical data may be used to represent a polyhedron and the use of all three together is highly redundant. However, a minimal representation of a polyhedron is not necessarily the best choice. Some redundancy may be justified.

Table 3.1.1.3.1-1. *Mathematical Representations of the Components of an Object*

Topological Element	Geometric Type	Equation for Element in Minimal Representation	Number of values that must be stored	Equation for Element in Homogeneous Representation	Number of values that must be stored
vertex	point	(x, y, z)	3	$\vec{V} = [x, y, z, m]$	4
face	plane	$ax+by+cz+1=0$	3	$\vec{P} = [a, b, c, k]^t$ $\vec{V} \cdot \vec{P} = 0$	4
edge	line	$x=(y-y_0)/a=(x-x_0)/b$	4	$L = \begin{bmatrix} \ell_{11} \cdots \ell_{14} \\ \ell_{21} \cdots \ell_{24} \end{bmatrix}$ $\vec{V} = [t\ 1] \cdot \vec{L}$	8

For complex manipulations and transformations in 3-space, it is common to use a mapping of the objects into a homogeneous representation in 4-space (Figure 3.1.1.3.1-1. Homogeneous coordinates provide a uniform formulation for translation, rotation, and scaling operations. The extra degree of freedom in the plane representation allows one to easily distinguish the inside from the outside of an object. For example, if $\vec{P}$ is a four element column vector representing a planar face, then the point represented by vector $\vec{V}$ will be on its inside or outside according to the sign of $\vec{V} \cdot \vec{P}$.

One disadvantage of a homogeneous representation is a reduction in the efficiency of storage.

3.1.1.3.2 Topological Representations

Within the topology of a polyhedron, there are nine possible types of relationships between the pairs of the three types of components (Figure 3.1.1.3.2-1). Storing all of these relations is highly redundant. In fact, one relation type is sufficient, and all others can be derived from it. Here again we have the usual tradeoff between computation and storage. The more information that is stored in a data base, the less that must be computed later. For vector graphics it is necessary to have one of the relationships v:{v}, e:{v}, or f:{v}, in order to know how many vertices are joined. Shape operations are facilitated by the ring of faces surrounding a vertex v:{f}. Euler number calculations and curved surface operations are aided by the adjacency among faces, e.g., f:{f}.

3.1.1.3.3 Storage of Three-Dimensional Objects

The choice of which geometric information to store depends upon display technique(s) and mix of data manipulation operations. Vertex coordinates are required when the data is to be plotted calligraphically. Surface normals or the equivalent face coefficients are required for raster scan display. Storing more than one type of information may be wasteful and can result in additional bookkeeping when the geometry of the face is altered.

It is sometimes desirable to store more than one type of topological relationship. Most systems store relationships in some sort of ring (circular linked list). The last component points to the first either explicitly or implicitly to complete the ring. It is possible to use doubly linked lists to facilitate the entry and deletion of components.

Topological relationships must satisfy a number of consistency conditions. Each edge must join exactly two vertices and two faces. Each face must be surrounded by an equal number (≥ 3) of vertices and edges. Each vertex must be surrounded by an equal number (≥ 3) of faces and edges. The number of faces F, vertices V, and edges E must satisfy the Euler equation

$$F + V - E = 2 - 2H,$$

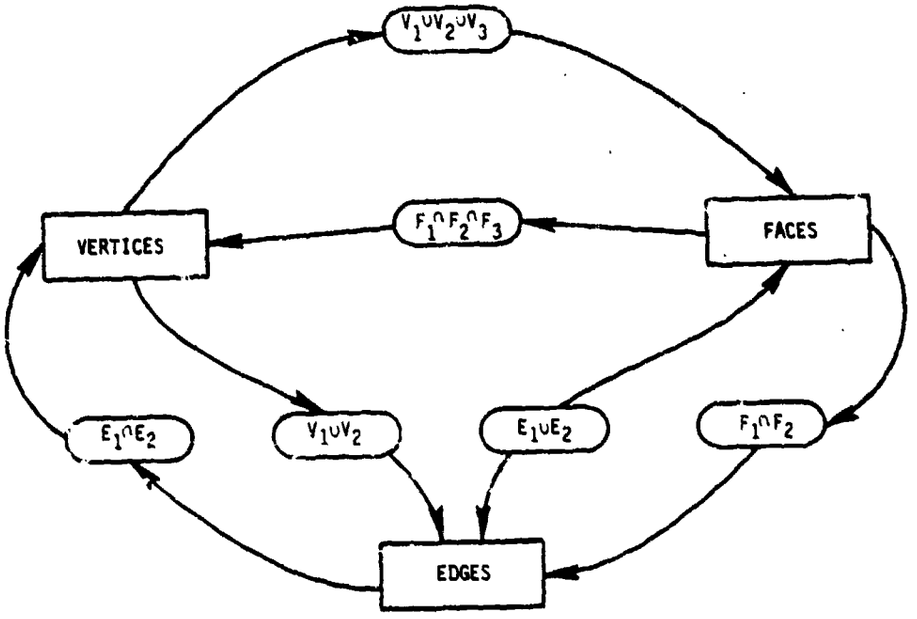


Figure 3.1.1.3.1-1. Six Conversions between Three Geometric Components

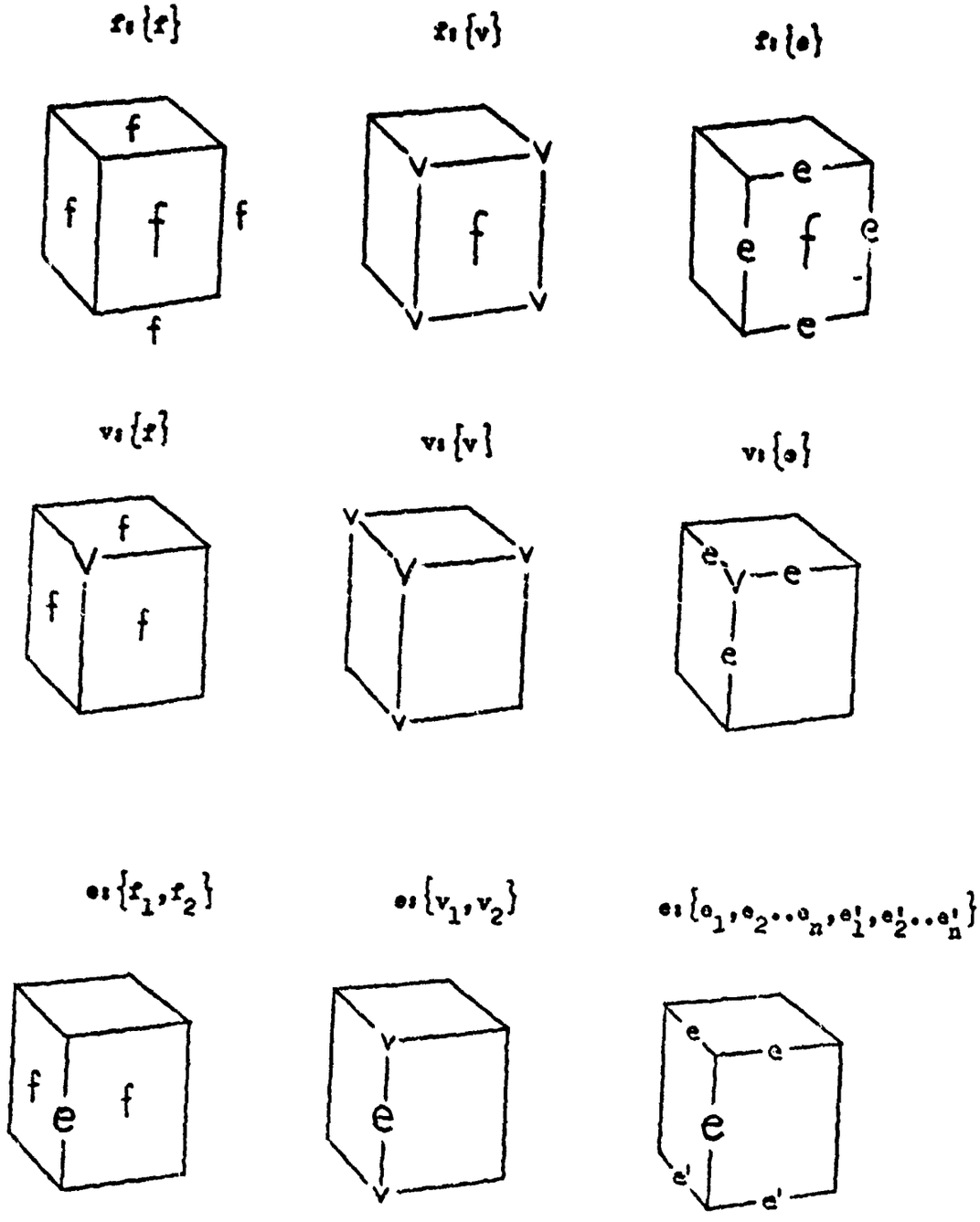


Figure 3.1.1.3.2-1. Nine Topological Relations between the Components of a Polyhedron

where H is the number of holes in the object. For graphic applications, usually H=0. An example is shown in Figure 3.1.1.3.3-1.

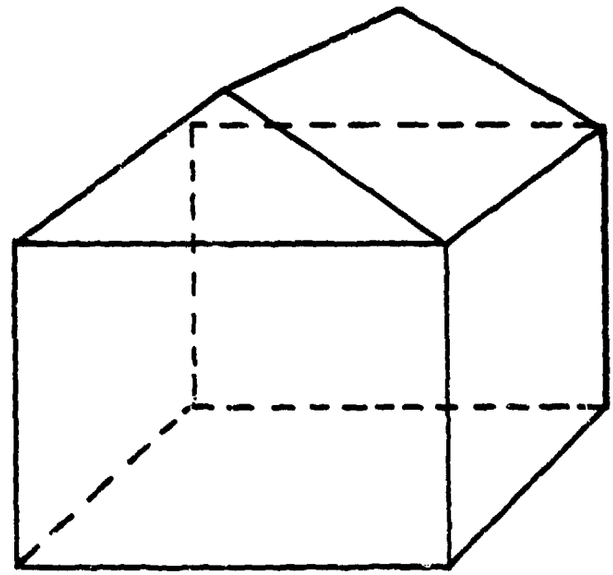
Satisfying the Euler equations is a necessary but not sufficient condition for a polyhedral surface to enclose volume. Moebius defined the orientation of a face in terms of the ordering of the edges and vertices around its perimeter, e.g., a clockwise ordering when viewed from outside the object. For a polyhedral surface to enclose volume, each edge must be traversed once in both directions. By ordering the elements in a corresponding way in a shape representation, one can make use of this property to distinguish inside from outside, as an alternative or in addition to the use of face normals.

The practical relevance of the Euler and Moebius conditions is that they may be used to ensure that shapes are well-defined and consistent. However, for certain applications, it may be desirable to relax these constraints. For example, one may wish to represent incomplete bodies such as open shells.

A general strategy for reducing the storage requirements for data is to identify the redundancies within it. Data that is similar and repeated throughout the data base may be stored once in common form and retrieved when needed. That is, storage may be saved by accepting some amount of additional computation to retrieve and process data each time it is needed. Usually, in computer graphics, the alternatives regarding what information to store and what to compute are significant, but tradeoffs are resolved intuitively or by experience. However, more precise forms of analyses are possible. These involve tradeoffs between costs of collections of redundant information and storage requirements of each, costs of alternative computation sequences, and upper bounds on scene display time.

One common form of redundancy in graphical data bases is the redundancy of objects. The same object may be required in several different locations. Each instance does not require a separate definition. Instead, the object can be represented once, with a spatial transformation used to specify each copy. The transformation will specify a translation, rotation, and scaling.

Carrying this approach to a more basic level provides the user with representations of frequently used classes of shapes. These shapes may be used as building blocks for constructing more elaborate objects. A particular shape may be stored or computed. If the shape is computed, the user must specify a few parameters from which the system computes the required data. For example, a cylinder may be specified by radius, height, and color.



NUMBER OF FACES = F = 9

NUMBER OF EDGES = E = 17

NUMBER OF VERTICES = V = 10

NUMBER OF HOLES = H = 0

EULER EQUATION

F+V-E+2H=9+10-17+0=2

1EA523A

Figure 3.1.1.3.3-1. Euler Equation Illustration

3.1.2 POLYGON VISUAL ENRICHMENT BY TEXTURING

The simulation of sensor scenes puts much heavier burdens on storage and processing of geometric data than general computer graphics applications. From the discussion in the preceding paragraphs, it is apparent that low level flight simulation greatly increases the storage problem if enough visual detail is provided to provide realistic perception of the scene. This problem can be greatly alleviated by coloring polygons with visual texture patterns which provide greatly enhanced perspective cues without requiring extra edge storage. Many rich visual texture patterns can be represented by a small number of mathematical parameters. Such patterns can be mapped onto polygons at some increase in display processing cost. The parameters can be adjusted to yield patterns which match the visual characteristics of plowed fields, forests, waves, and cities. The following paragraphs describe mathematical properties of texture patterns. Applications to scene texture generation may be found in C.11, C.23, G.32 and V.30.

3.1.2.1 Introduction

A variety of properties have been used to characterize image textures. Several investigators analyzed textures by means of the Fourier transform. Rosenfeld and Troy (V.26) suggested a measure based upon local edge density. Haralick, Shanmugin, and Dinstein (V.14) extracted a number of parameters from a texture's concurrence matrix. Davis, Johns, and Aggarwal (V.8) recently generalized the concept with improved results. Galloway (V.11) analyzed textures by means of frequency of runs of constant gray level. Schachter, Davis, and Rosenfeld (V.27) used a norm known as total variation to quantify texture coarseness. Tamura, Mori, and Yamawaki (P.22) list six basic texture properties perceived by human observers (coarseness, contrast, directionality, linelikeness, regularity, and roughness). They suggest formulae for computing these properties. The following paragraphs describe several texture models and discuss their mathematical representation.

3.1.2.2 Random Fi

To every point $x \in E^2$, attach a random variable $Z(x)$. The mean function of $Z(x)$ is defined as:

(1) $m(x) = E[Z(x)]$.

The covariance function of $Z(x)$ is given by

(2) $C(x, y) = E[Z(x)\ Z^*(y)] - m(x)m^*(y)$,

where '*' denotes complex conjugate.

The variogram is defined as

$$V(x,y) = E[Z(x) - Z(y)]^2. \tag{3}$$

The correlation function of $Z(x)$ is given by

$$\rho(x,y) = \frac{C(x,y)}{[C(x,x)C(y,y)]^{1/2}}. \tag{4}$$

The cross-correlation between two fields Z_1 and Z_2 is defined as

$$C_{12}(x,y) = E[Z_1(x)\ Z_2^*(y)] - m_1(x)m_2^*(y). \tag{5}$$

The covariance and correlation functions defined in (2) and (4) are sometimes given the prefix "auto" to distinguish them from "cross" functions in the literature.

A random field is (wide sense) <u>stationary</u> if its mean is a constant and its covariance function depends only on (x-y), i.e., $C(x,y) = C(x-y)$, $\rho(x,y) = \rho(x-y)$, and $V(x,y) = V(x-y)$. In terms of Euclidean motions, stationarity implies invariance under translation. Furthermore, if stationarity holds, the relationship

$$C(x-y) = C(0)\rho(x-y) \tag{6}$$

exists between the two classes of functions.

A random field is (wide sense) isotropic if its covariance function is invariant under all rotations about a fixed point. Isotropy is most easily defined under a polar coordinate system (r,θ), with the origin used as the point of rotation. If isotropy holds, then

$$E[Z(r,\theta)\ Z^*(r_o,\theta_o)] = E[Z(r,\theta+\delta)\ Z^*(r_o,\theta_o+\delta)], \tag{7}$$

for all δ, where angular measures are modulo 2π. By setting $\delta = -\theta_o$ in the above equation, it becomes apparent that the covariance function depends only on $\theta-\theta_o$.

A random field is called homogeneous if it is stationary and isotropic. The notation used to describe random fields has not been standardized. The terminology here follows that given by Wong (V.33). Homogeneity implies invariance under all rigid Euclidean motions. If homogeneity holds $C(x, y) = C(d)$, $p(x, y) = p(d)$, and $V(x, y) = V(d)$; where d is the Euclidean distance between points x and y.

A discrete real scalar homogeneous random field will be used as the model of spatial variation.

3.1.2.2.1 Gaussian Random Fields

Consider a texture region ACE^2 of unit area. Let G(x) denote the gray level at point $x \epsilon A$. A real field $\{G(x) | x \epsilon A\}$ is a Gaussian random field if for every finite set of points $x_1 \epsilon A$, the corresponding gray levels $G(x_1)$ are jointly Gaussian random variables. A homogeneous Gaussian random field is completely characterized by its mean and covariance.

The field reduces the Gaussian white noise if $\rho(d) = \rho(0)\,\Delta(d)$, where $\Delta(d) = 1$ when $d = 0$, and $\Delta(d) = 0$ otherwise. A white noise texture has no coarseness since each pixel is colored independently. The business of a texture is often defined as the frequency of transitions from light to dark. Variance is a measure of business.

Let us investigate a particular Gaussian random field for which $\rho(d>0)$ is not equal to zero. The class of textures that we will consider can be produced by sums of periodic waveforms. Textures of this type can be described by a contour map. A contour is a line on the map that connects neighboring points of equal gray level. The difference in gray levels between adjacent contour lines is the contour interval of the map. Closely spaced contour lines indicate a high gray level gradient: sparseness or the absence of contours indicates a mild gray level drift or constancy.

One property of the texture within A is the total length of contour lines. Let $L \epsilon \ell$ denote a particular contour level. Let α denote arc length. The mean <u>roughness</u> E[R] of the texture can be written as:

$$E[R] = E\left[\sum_{L \epsilon \ell} R_L\right] \quad \sum_{L \epsilon \ell} E[R_L]$$

where $R_L = \alpha\{x \epsilon A : G(x) = L\}$.

Switzer (G.35) has derived E[R] for a homogeneous Gaussian random field. He obtains the result

$$E[R_L] = \tfrac{1}{2}|\rho''(0)|^{\frac{1}{2}} \exp(-\tfrac{1}{2}L^2),$$

where G is scaled to have zero mean and unit variance and ρ'' denotes the second derivative of the correlation function. When contour intervals are equally spaced

$$E[R] \simeq \varepsilon \int_{-\infty}^{\infty} R_L \, dL = \tfrac{1}{2} |\rho''(0)|^{\frac{1}{2}} \int_{-\infty}^{\infty} \exp(-\tfrac{1}{2}L^2) \, dL = |\tfrac{1}{2}\pi\rho''(0)|^{\frac{1}{2}}.$$

A more common measure of roughness is the mean magnitude of the image gradient $\mathcal{G}$, where

$$\mathcal{G}(x) = \left[\left[\frac{\partial G(x)}{\partial u} \right]^2 + \left[\frac{\partial G(x)}{\partial v} \right]^2 \right]^{\frac{1}{2}}.$$

Under the given assumptions $\partial G(x)/\partial u$ and $\partial G(x)/\partial v$ are themselves Gaussian random variables with zero mean and variance $\rho''(0)$. The square of a Gaussian random variable has a x^2 distribution. Thus, the magnitude of the gradient is proportional to the mean of a x^2 random variable with two degrees of freedom. Switzer (G.35) obtains

$$E[R] = |\tfrac{1}{2}\pi\rho''(0)|^{\frac{1}{2}}.$$

It is seen that both derivations of a roughness measure yield the same result. Thus, $|\tfrac{1}{2}\pi\rho''(0)|^{\frac{1}{2}}$ is an appropriate measure for the roughness of this class of textures.

The fact that "roughness" characterizes some property of this texture does not necessarily imply that coarseness or patchiness are inappropriate descriptors. However, consider the following intuitive argument claiming that they are. In a famous series of papers, Longuet-Riggins modelled the height of ocean waves by sums of long crested waveforms (e.g., see (V.17)). His model is very similar to the one described above. Thus, turbulent seas are always described as rough; never as busy, patchy, or coarse.

3.1.2.2.3 Random Mosaics

Mosiacs in art are designs or pictures made by glueing colored objects onto a flat surface. The Romans classified mosaics into several categories, according to the kinds and forms of materials used. "Opus tessellatum" referred to a motif of stone squares arranged in a simple geometric pattern. This has given rise to the modern term "tessellation" denoting the covering of the plane by nonintersecting cells.

Random mosaics may be used to depict homogeneous textures. To be more explicit, a random mosaic will be defined by a two-step construction.

(1) **Randomly tessellate a planar region A, of unit area, into cells. Consider only tessellations composed of convex cells. This tessellation must have the property that the probability that two randomly chosen points fall within the same cell depends only upon the distance between them.**

(2) **Independently assign one of m colors to each cell according to the fixed set of probabilities $p_1, \ldots, p_m$;**

$$\sum_{i=1}^{m} p_i = 1.$$

By this process, partition A into subregions $A_1, \ldots, A_m$;

$$\bigcup_{j=1}^{m} A_j = A,$$

where A_j is defined to be the union of all cells of color j.

The partitioning of A is the realization of a random process with the following stationary and transition probabilities (V.18):

(1) For all $s \in A$, $Pr(s \in A_i) = p_i$ for $i=1,2,\ldots,m$.

(2) For all $(s,s') \in A$, with distance $d=|s-s'|$ between them,

$Pr(s' \in A_i \mid s \in A_j) = P_{ij}(d) = p_i\{1-W(d)\} + \zeta_{ij} W(d)$ for $i,j=1,2,\ldots,m$.

$W(d)$ is the probability that any two randomly chosen points a distance d apart are both in the same cell of the tessellation. ζ_{ij} is the Kronecker delta.

By saying that a cell is of color-i below, it is meant that the points within it have the distribution $N(\upsilon_i, \sigma_i^2(d))$. Two or more contiguous cells of the same color will be said to form a <u>patch.</u>

The variogram for a random mosaic is given by (G. 32) as

$$V(d) = \sum_{ij}\sum\{\sigma_i^2(d)+\sigma_j^2(d)+(\upsilon_i-\upsilon_j)^2\}p_j P_{ij}(d)$$

$$= \sum_{i\neq j}\sum\{\sigma_i^2(0)+\sigma_j^2(0)+(\upsilon_i-\upsilon_j)^2\}p_j p_i\{1-W(d)\}$$

$$+ \sum_i 2p_i\sigma_i^2(d)W(d) + \sum_i 2p_i^2\sigma_i^2(d)\{1-W(d)\}.$$

The first term of the above equation deals with inter-region relationships, the second term with intraregion relationships, and the third term with intracell relationships. Note that if there is no dependence across intraregion cell boundaries then $\sigma_i^2(d)$ in the second term of the equation becomes $\sigma_i^2(0)$, if there is no dependence between points within a cell then $\sigma_i^2(d)$ in the third term of the equation becomes $\sigma_i^2(0)$.

Matern (V. 18) has shown that certain properties of a mosiac can be derived from the first derivative of the variogram and correlation function:

$$B_p = \lim_{d\to 0}\frac{V(d)-V(0)}{d} = \lim_{d\to 0}\{V(d)/d\} = V'(0)$$

and

$$B_c = -\lim_{d\to 0}\frac{\rho(d)-\rho(0)}{d} = -\lim_{d\to 0}\frac{\rho(d)-1}{d} = -\rho'(0).$$

B_ρ is a measure of the <u>patchiness</u> of a mosiac, B_c is a measure of <u>coarseness</u>. These measure s are best understood by considering an example.

<u>Poisson Line Mosaic:</u> Consider a tessellation composed of lines in the plane with random positions and orientations. Such a system when derived by the following process possesses the fundamental properties of stationarity and isotropy. A Poisson process of intensity τ/π chooses points (θ,b) in the infinite rectangular strip $\{0\leq\theta<\pi, -\infty<b<\infty\}$. Each of these points may be used to construct a line in the plane of the form

$$x\cos\theta + y\sin\theta - b = 0,$$

where b is the signed distance to an arbitrarily chosen origin. One may use this process to tessellate any finite region A into cells. Now complete the mosaic by independently assigning one of m colors to the cells according to the fixed set of probabilities $p_1,\ldots,p_m$. These colors will refer to distributions $N(\upsilon_i,\sigma_i^2)$, $i=1,2,\ldots,m$.

Now suppose that we drop a Buffon needle onto the mosaic. Let ℓ_c denote the length of intersection of the needle with a cell of the mosaic, and let ℓ_a denote the length of intersection with a patch of color-a. Pielou (V.22) has shown that ℓ_c has an exponential distribution with mean$(\pi/2\tau)$ and $\ell_a$has an exponential distribution with mean $\pi/2\tau(1-p_a)$. Hence, the expected number of times that a needle of unit length intersects cell boundaries is $\sum_{i=1}^{m}(2\tau/\pi)p_i(1-p_i)$.

Let $\upsilon=\sum_i p_i\upsilon_i$. The correlation function for this texture is:

$$\rho(d) = \frac{C(d)}{C(0)} = \frac{C(d)}{\sigma^2} = \frac{1}{\sigma^2}\{\sum_i(p_i\upsilon_i \sum_j \left[P_{ji}(d)\upsilon_j\right])-\upsilon^2\}$$

$$= \frac{1}{\sigma^2}\{\sum_i\left(p_i\upsilon_i \sum_{j\neq i}(p_j\left[1-W(d)\right]\upsilon_j)+(p_i+(1-p_i)W(d))\upsilon_i\right) -\upsilon^2\}$$

$$= \frac{1}{\sigma^2}\{\sum_i(p_i\upsilon_i(\upsilon\left[1-W(d)\right]+\upsilon_i W(d)) -\upsilon^2\}$$

$$= \frac{1}{\sigma^2}\{\upsilon\left[1-W(d)\right]\sum_i(p_i\upsilon_i)+W(d) \sum_i(p_i\upsilon_i^2)-\upsilon^2\}$$

$$= \frac{1}{\sigma^2}\{\upsilon^2\left[1-W(d)\right]+W(d)\ E\left[\upsilon_i^2\right]-\upsilon^2\}$$

$$= \frac{1}{\sigma^2}\{E\left[\upsilon_i^2\right]-\upsilon^2\}W(d) \quad = \frac{\sigma^2}{\sigma^2}W(d) = W(d) = \exp(-2\tau d/\pi).$$

The coarseness of this texture is $B_c = -\rho'(0) = (2\tau/\pi)$. Thus, the first derivative of the correlation function supplies information about the mosiac's underlying microstructure.

For this example, since it is assumed that $\sigma_i^2(d)=\sigma_i^2(0)$, the variogram given in (V.26) becomes

$$V(d) = \sum_i\sum_j\{\sigma_i^2+\sigma_j^2+(v_i-v_j)^2\}p_jp_i\{1-W(d)\}+\sum_i 2\sigma_i^2p_iW(d)$$

$$= \sum_i\sum_j\{\sigma_i^2p_jp_i\left[1-W(d)\right]\} + \sum_i\sum_j\{\sigma_j^2p_jp_i\left[1-W(d)\right]\}$$

$$+ \sum_i\{2\sigma_i^2p_iW(d)\} + \sum_{i\neq j}\sum\{(v_i-v_j)^2p_ip_j\left[1-W(d)\right]$$

$$= 2\sum_i\{p_i\sigma_i^2\} + \sum_{i\neq j}\sum\{p_ip_j\left[1-W(d)\right](v_i-v_j)^2.$$

Thus

$$V'(0) = \sum_{i\neq j}\sum\{(2\tau/\pi)p_ip_j(v_i-v_j)^2\}.$$

This example is simplified by considering a two-color mosaic. Cells are independently colored black ($v_1=1$) with probability p_1 or white ($v_2=0$) with probability $p_2 = 1 - p_1$. The variogram now reduces to

$$V(d) = 2p_1\sigma_1^2 + 2p_2\sigma_2^2 + 2p_1p_2\{1-W(d)\}.$$

$$V'(0) = \sum_{i=1}^{2}\{(2\tau/\pi)p_i(1-p_i)\}$$

Thus for this simple case Bp equals ℓp, the expected number of patch intercepts on a randomly dropped test line of unit length.

In summary, mosaic textures are generated by a two-stage process:

(1) a partitioning of the plane into cells; and

(2) a coloring of the cells.

B_c is a measure of the intensity of the first stage of the process. B_p appears to offer a reasonable summary of the statistics of the second stage.

3.1.2.2.4 Bombing Models

Random two-color patterns can be formed by "bombing" processes. The bombs are geometric figures that are dropped onto the plane. The sizes and shapes of these figures are fixed, but their positions and orientations are random. The foreground (union of the dropped figures, "bombed out" region) is assigned one color (say $N(\upsilon_1,\sigma_1^2)$) and the background a second color (say $N(\upsilon_2,\sigma_2^2)$).

A bombing texture is in some sense less symmetric than a two-color mosaic, since its foreground and background are formed by different processes. Because this texture is not cellular, $B_c = \rho'(0)$ has no obvious significance. However, patchiness can still be characterized as a measure of the border per unit area between the foreground and background.

For a circular bombing process of intensity λ, the variogram is given by

$$V(d) = \sum_{i=1}^{2} \sum_{j=1}^{2} \{\sigma_i^2 + \sigma_j^2 + (\upsilon_i-\upsilon_j)^2 p_j P_{ij}(d)\};$$

where p_1 is the probability that a point on the plane is covered by a circle, $p_1=1-\exp(-\lambda\pi r^2)$, $p_2=1-p_1$. Switzer (G.35) gives

$$p_1 P_{11}(d) = (2p_1-1) + p_2^{H(d/r)}, \text{ where}$$

$$H(d/r) = \begin{cases} 1 + (2/\pi)\{\frac{d}{2r}\left[1-\left(\frac{d}{2r}\right)^2\right]^{\frac{1}{2}} + \sin^{-1}(d/2r)\} & \text{for } d \leq 2r \\ 2 & \text{for } d>2r. \end{cases}$$

The remaining terms of the equation for the variogram are given by

$$P_{12}(d)=(p_1/p_2)P_{21}(d), \quad P_{22}(d)=1-P_{12}(d), \quad P_{21}(d)=1-P_{11}(d).$$

Therefore,

$$V(d) = 2p_1\sigma_1^2 + 2p_2\sigma_2^2 + 2p_1P_{21}(d)(\upsilon_1-\upsilon_2)^2$$

$$= 2p_1\sigma_1^2 + 2p_2\sigma_2^2 + 2(p_2 + p_2^{H(d/r)})(\upsilon_1-\upsilon_2)^2.$$

Then

$$V'(d) = \frac{\partial}{\partial d}\left[p_2^{H(d/r)}(v_1-v_2)^2\right] = (v_1-v_2)^2\frac{\partial}{\partial d}\left\{\left[\exp(-\lambda\pi r^2)\right]^{H(d/r)}\right\}$$

$$= (v_1-v_2)^2\frac{\partial}{\partial d}\left\{\exp\left[-\lambda\pi r^2 H(d/r)\right]\right\}.$$

$$B_p = V'(0) = 2(v_1-v_2)^2\lambda\pi r\{\exp(-\lambda\pi r^2)\} = 2p_2\lambda\pi r(v_1-v_2)^2.$$

Thus, patchiness is related to the intensity of the bombing process and the contrast between the foreground and background.

3.1.2.3 Discussion

Many measures have been developed to quantify the structural properties of image texture, e.g., "business", "patchiness", "coarseness", and "roughness". These terms have been made more precise by relating them to the stochastic geometry of a texture's microstructure. Furthermore, this entire set of measures should not be indiscriminately applied to any texture. The topology of a texture surface actually prescribes the use of an appropriate subset of these. A descriptor set for a texture may also include other properties, such as mean gray level and measures of inhomogeneity.

3.1.2.4 Evaluation of Texture Efficiency

Figure 3.1.2.4-1 is an image of sixteen texture patterns used in an advanced General Electric Flight Simulator. In left-to-right, top-to-bottom order, these are used to represent the surface material cover listed in the left column of Table 3.1.2.4-1. The view in Figure 3.1.2.4-1 is seen from 6000 feet above the patterns, with a field of view of 40 degrees. Thus, each texture patch is about 1000 feet on one side which is about the size of a typical terrain triangle for the finest level of detail on the simulator. Figure 3.1.2.4-2 is a perspective view of the texture used for swamps. Note the vivid perception of depth caused by the perspective compression of the pattern with distance.

The edge savings gained by using texture can be evaluated by estimating the number of edges required to mimic each pattern shown in Figure 3.1.2.4-1. These numbers appear in the middle column of Table 3.1.2.4-1. Dividing by the area of each texture patch yields the edge density per square nautical mile which appears in the right column of Table 3.1.2.4-1. These numbers represent the number of edges that do not have to be modeled to achieve visual richness via texturing.

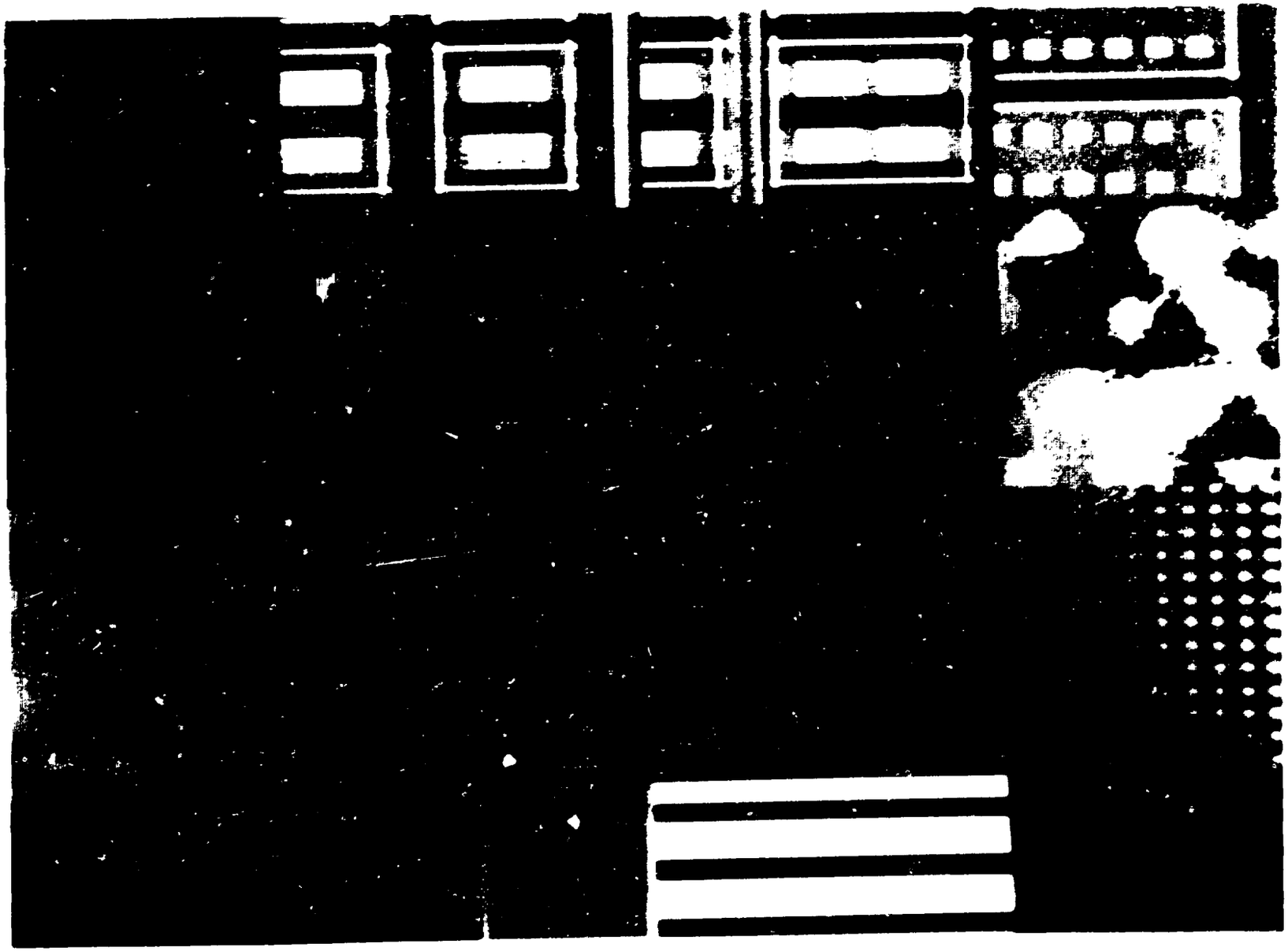


Figure 3.1.2.4-1. Texture Patterns

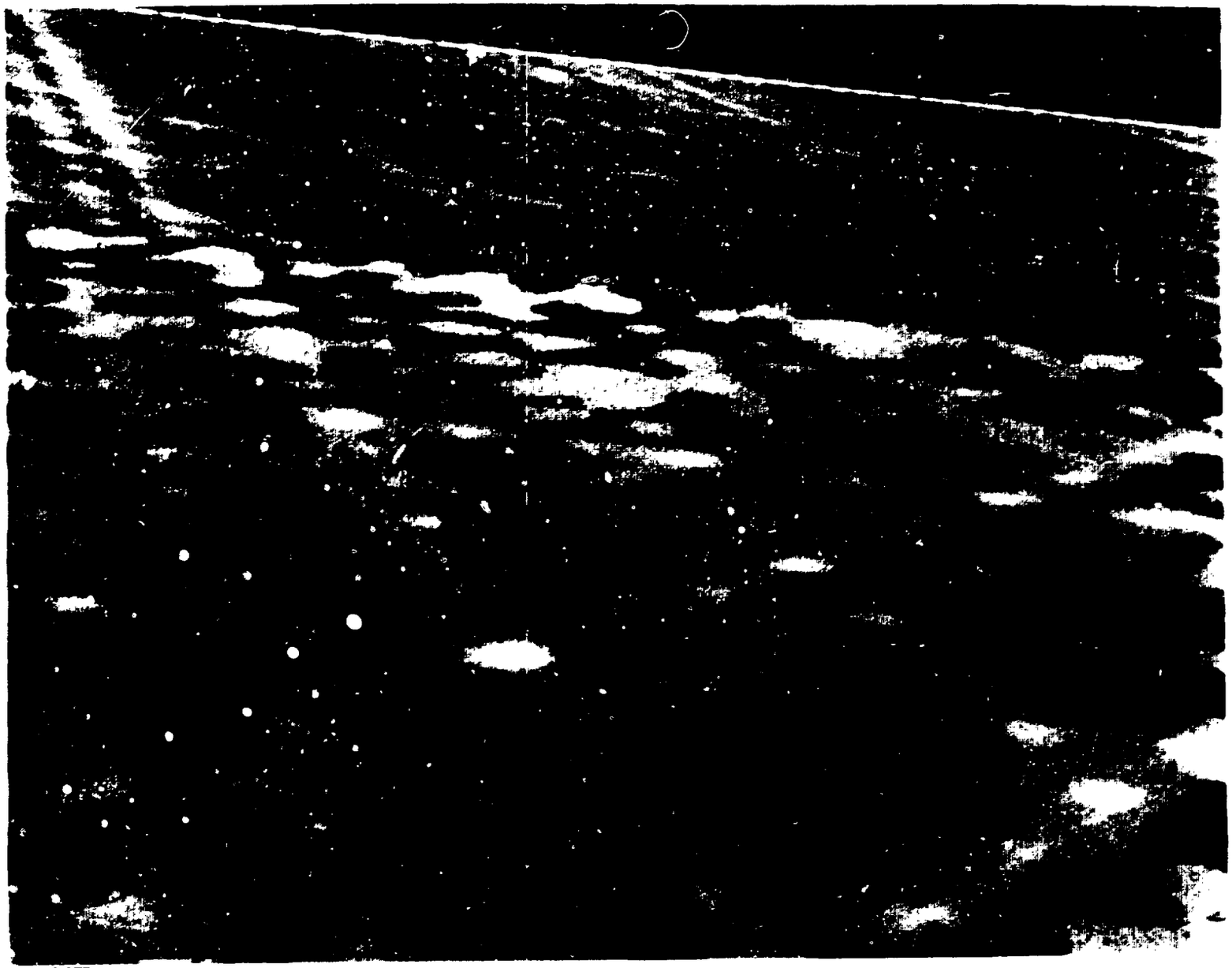


Figure 3.1.2.4-2. Perspective Texture Representing Swamplands

Table 3.1.2.4-1. Edge Equivalents of Texture Pattern

	Surface Material	Edges Per Typical Terrain Face	Edges per nm^2
1.	Water	200	7,384
2.	Industrial	26	960
3.	Urban	60	2,215
4.	Suburban	120	4,430
5.	Runway Surface	5304	195,811
6.	Normal Soil	256	9,451
7.	Desert	35	1,292
8.	Swamp	42	1,551
9.	Rocky Surface	52	1,920
10.	Plowed Field	79	2,917
11.	Mud/Tidal Flat	23	849
12.	Orchard	1014	37,435
13.	Forest	1350	49,839
14.	Farmland	169	6,239
15.	Runway Stripe (10,000 feet)	192	1,073
16.	Cloud Layer	23	849

Recall that scene edge density was 265 edges per nm^2 in the very high detail target areas discussed in paragraph 2.1.2. The use of forest texture provides nearly a 50,000 edge equivalent for the same area according to Table 3.1.2.4-1. Thus, texture improves storage efficiency by a factor of two hundred-to-one for equivalent visual richness. For rather featureless desert areas, the ratio is only about five-to-one. Even this smaller number represents a considerable improvement in scene generation system storage required to represent equivalent visual content using texture.

While the preceding analysis concerns only a single level of detail, textures for flight simulators are designed at multiple levels, based on the same kinds of resolution change as described for terrain level of detail in paragraph 2.1.4. Thus, texture edge equivalent density changes by a factor of two for each level of detail, and the efficiency ratios derived in the preceding paragraph hold for all levels of detail.

3.1.3 POLYGON DATA BASE DISPLAY

Several software packages for generating scenes from polygonal data bases with a general-purpose digital computer, are described below.

3.1.3.1 Brigham Young University MOVIE System

The input data to the MOVIE system can consist of planar faces, solids, or contour drawings. Routine MOVIE is the heart of the system. It is the display program for triangular and quadrilateral elements. The executable program consists of three FORTRAN source files: MOVIE. FOR, HIDDEN. FOR, and DEVICE. FOR compiled and loaded together.

MOVIE. FOR is the interactive command processor. HIDDEN. FOR does the hidden surface removal calculations, using the Watkins algorithm. DEVICE. FOR takes care of display activities.

Translation and rotation of the model in any direction or about any one of the cartesian coordinates is possible using rigid body commands. Other commands allow the selection of color and shading rules for continuous tone images.

3.1.3.2 General Electric Static Scene Generators

Several experimental software systems are used at General Electric to simulate aircraft flight scenes. They are designed for minicomputer simulation of CIG systems to evaluate new algorithms before committing to expensive hardware implementation. The time required to generate the image of a single frame in full color changes from a few minutes to over an hour, depending on scene richness and processing complexity. The general flow of data in these software systems is illustrated in Figure 3.1.3.2-1. Several scenes generated by such systems are shown in Figure 3.1.3.2-2, 3.1.3.2-3, 3.1.3.2-4 and 3.1.3.2-5.

Figure 3.1.3.2-4 is a reproduction of a CIG image of an airfield seen from several thousand feet up. The visible scene is composed of about 14000 edges, roughly half of which are contained in the taxiways and runways with their assorted markings. Many of these markings are too small to be seen from this elevation. A low altitude view of an airfield, including haze, is provided in Figure 3.1.3.2-5.

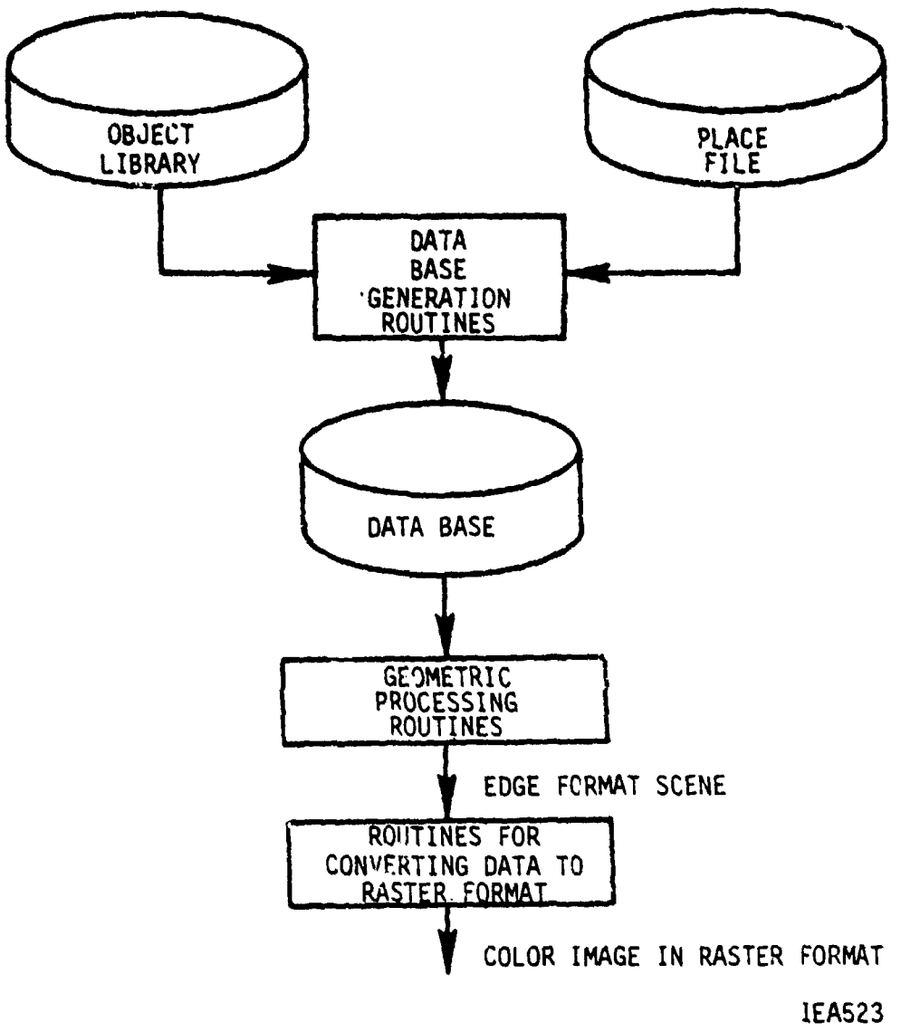


Figure 3.1.3.2-1. Scene Generation System

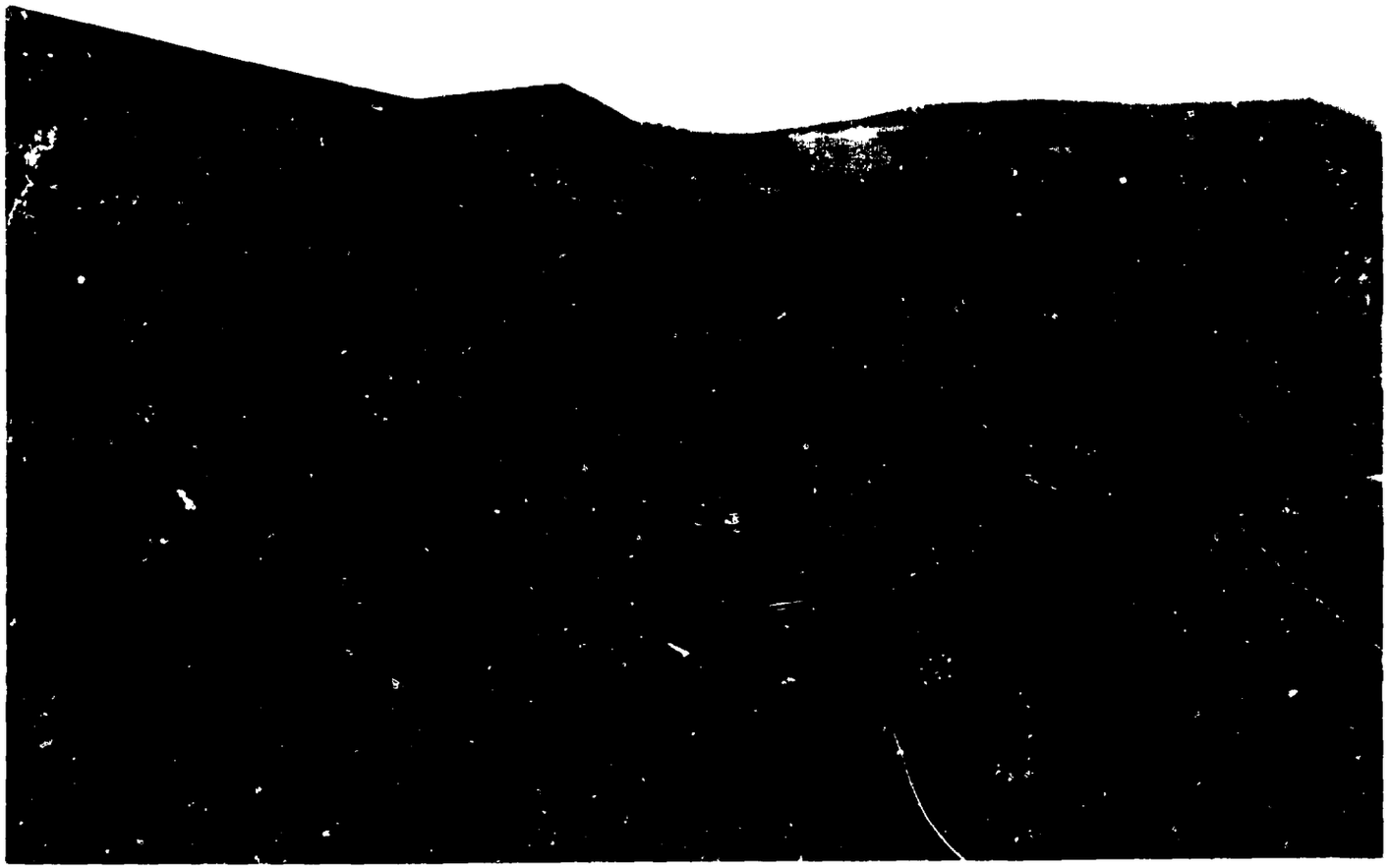


Figure 3.1.3.2-2. Textured Perspective Scene with Mountains and Lakes

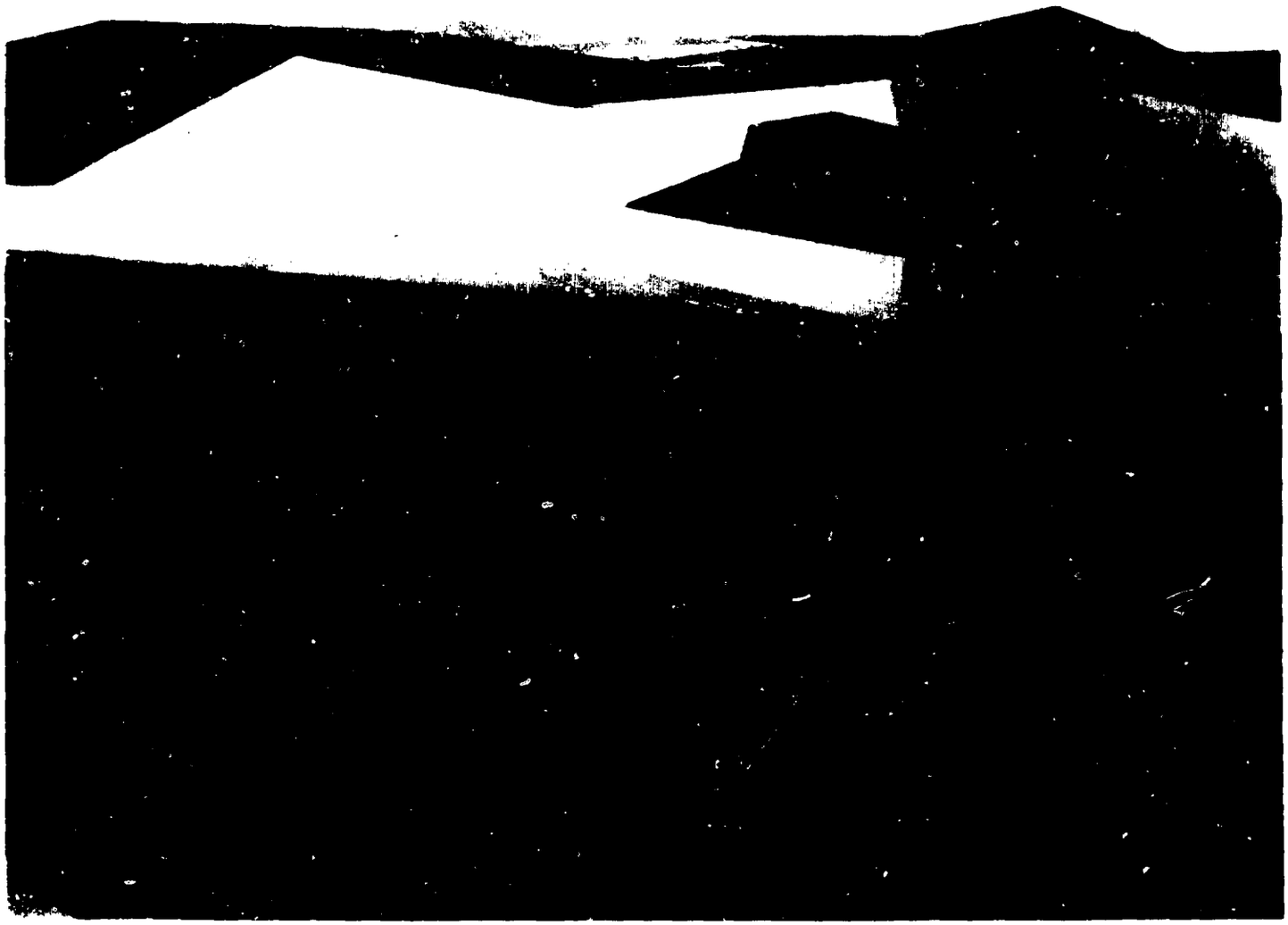


Figure 3.1.3.2-3. Textured Perspective Scene with Mountains

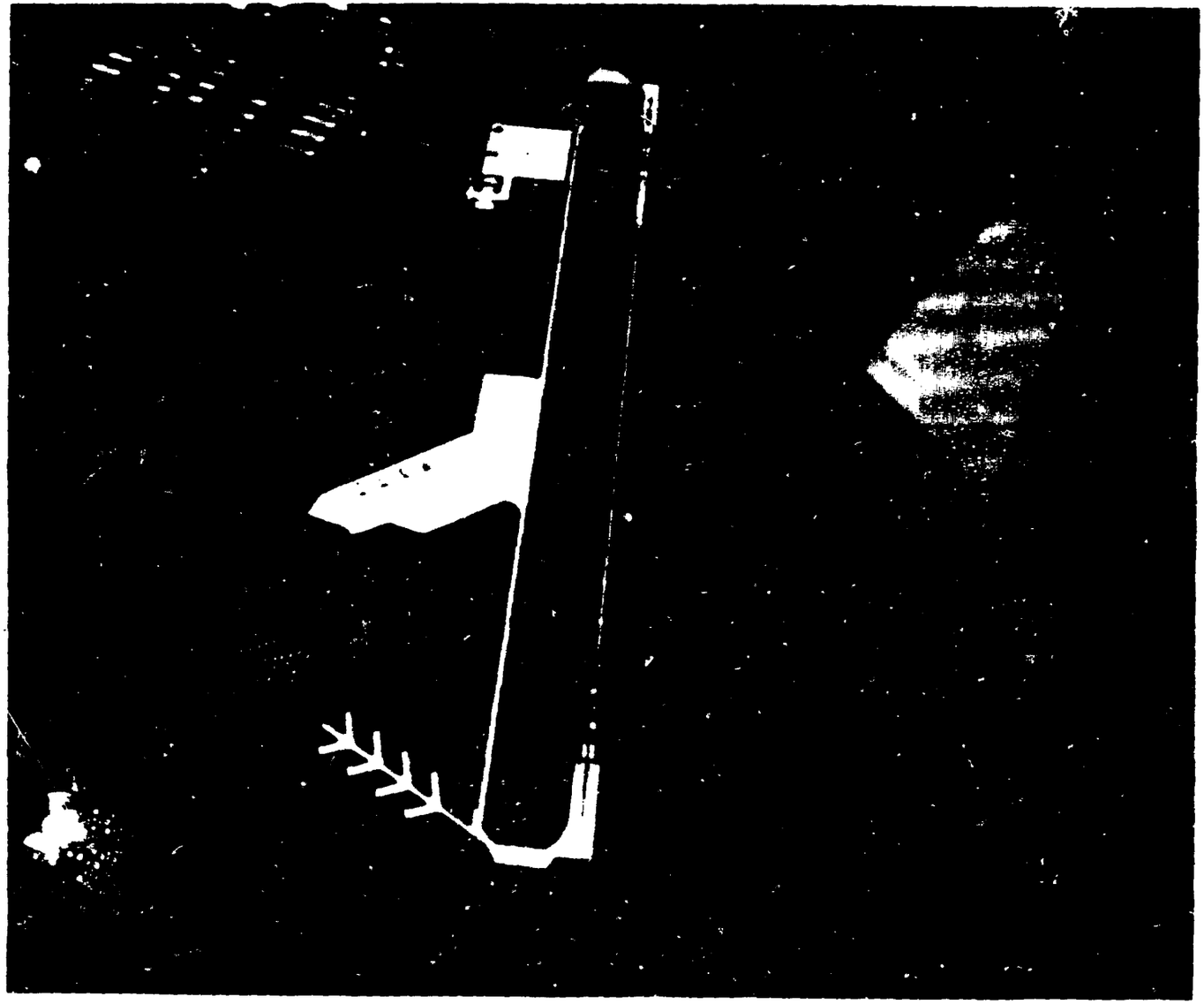


Figure 3.1.3.2-1. Textured Perspective Scene with Airfield

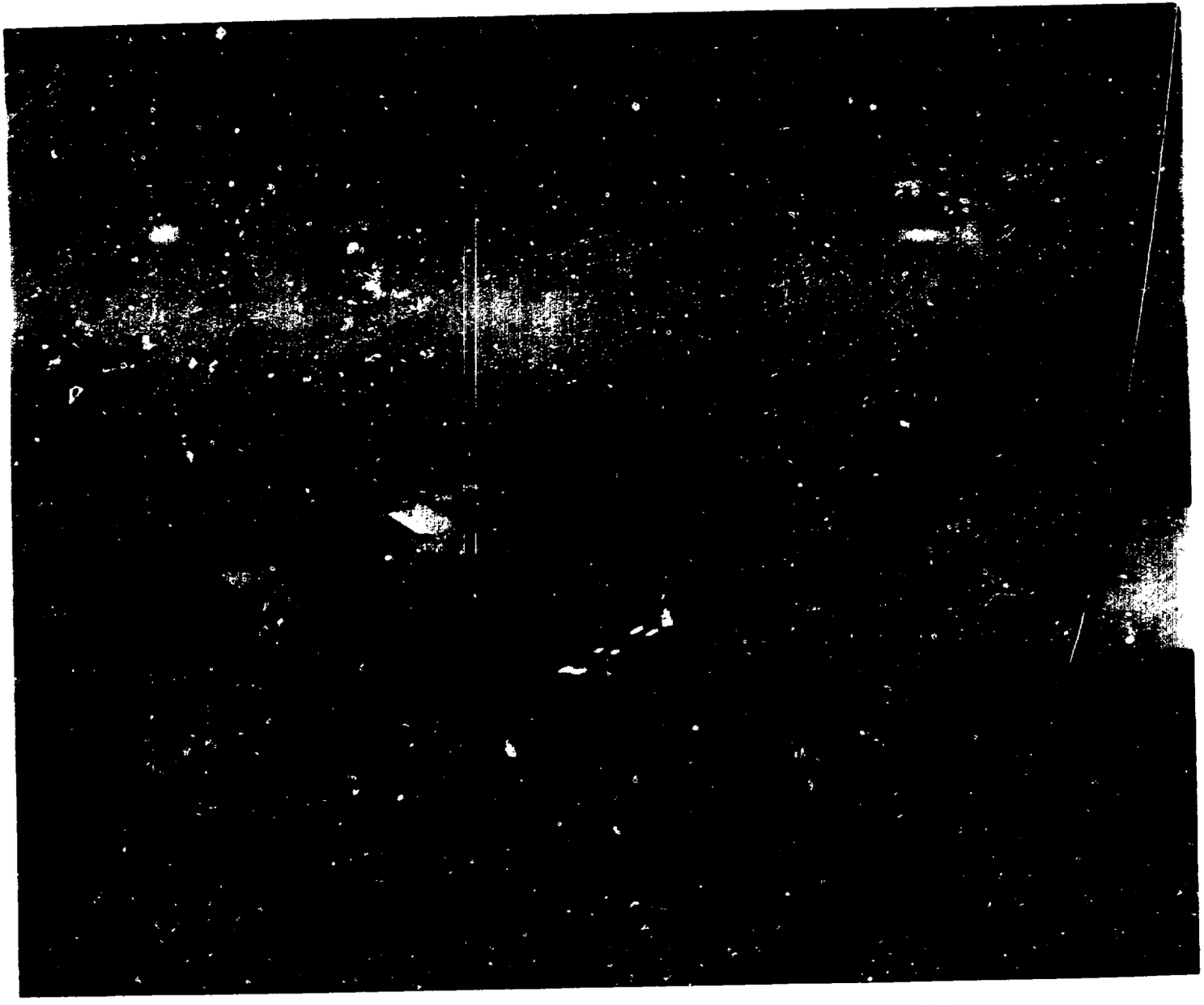


Figure 3.1.3.2-5. Low Altitude Airfield Scene with Texture and Haze

Figure 3.1.3.2-3 is a reproduction of a CIG image of triangulated DMA terrain. Although the data within the field of view technically contains about 350 edges, only a few dozen are visible. The remainder are principally terrain polygons which are hidden behind hills and ridges looming large in the foreground. This geometric situation is typical of low altitude flight scenes and emphasizes the value of texture in providing rich perspective cues on what would otherwise be featureless surfaces. The edge equivalence of the visible edges is approximately 800, more than ten times the number of visible polygon edges.

3.1.3.3 Mathematical Applications Group's SYNTHAVISION

SYNTHAVISION is primarily used for making animated cartoons for TV commercials. General Electric has obtained the rights to the package. Some General Electric locations have implemented this system on a VAX computer equipped with an array processor.

SYNTHAVISION constructs objects from generic solids (e.g., cylinders, cones, cubes, ellipsoids, etc.) These solids may be combined in any fashion to form more complex objects. Since SYNTHAVISION builds volumes from geometric shapes rather than vertices, scene definition is easier than for most other systems.

3.2 DIRECT DISPLAY OF GRID DATA BASES

3.2.1 INTRODUCTION

The representation of geometric data in terms of polygons has a long and fruitful history in computer graphics for several reasons. Among these are that relatively few data items (e.g., a list of vertices) can be used to represent large geometric entities (polygons and figures constructed from them). Additionally, the linear geometry of planes provides simple interpolating computations for generating crisp projected displays which are unambiguous in visual interpretation. Analytic geometry provides straightforward techniques for transformation of data corresponding to motion, scaling, or rotation without altering the underlying data structure. Computer graphics was first applied to generating displays of rather simple objects representing boxes, buildings and vehicles. As algorithms became more efficient and hardware performance improved, real-time computer graphics applications arose in flight training with the birth of visual flight simulation via CIG. In this application, terrain was represented by a few large planes plus a few polyhedral mountains or hills. More recently, terrain has been represented as a complex network of polygons which closely fit real world terrain. This has introduced some new data storage and processing problems which are beginning to offset the efficiency of polygon representation of terrain in visual and sensor simulation. The nature of these problems and the alternative of rendering displays directly from grid data are discussed below.

Visual flight simulators have approached the challenge of low level detection avoidance mission simulation by increasing terrain visual content per unit area. This is achieved by adding texture to polygons and increasing the fineness of polygonal representation of rough terrain. This results in very large data bases which must include an interlocking data structure to provide for decreasing the level of detail at extreme ranges to prevent system overload. Such processing overloads are often caused by large numbers of terrain polygons overlapping relatively few raster display lines near the horizon. Furthermore, the generation of the polygon network from source data becomes much more expensive as the network more closely approximates real terrain (C. 23). The source data for terrain simulation is usually the Defense Mapping Agency digital data base which consists of a regular terrain elevation grid, the outlines of planimetric data which characterize ground material, and points designating localized features such as towers.

Hardware processing costs per unit of image resolution increase with polygon based methods but decrease with grid based methods. Recent advances in VLSI and parallel algorithm design are tipping the balance dramatically in favor of grid based methods, particularly in low level flight environments. When the crossover point is reached within the next few years, a radical improvement will occur in scene generation hardware and image quality. Several historical disadvantages to grid techniques

are diminishing due to advances in VLSI and some new algorithms. One such disadvantage is that grid data bases require about twice as much storage as polygon data bases to represent terrain at any given elevation accuracy (D.11). A more fundamental problem is that very little work has been done in developing geometrically valid algorithms for rendering images directly from grid data because of the success of polygon representations in most other applications of computer graphics. Most grid rendering algorithms are variants of ray tracing; that is, each pixel is assumed to be an image of a ray projected from the viewpoint through the terrain surface. Some of the difficulties encountered include complex hidden parts calculations, ad hoc handling of sampling mismatches due to perspective effects, and the lack of geometrically valid interpolation between elevation points. A few successful approaches have only recently been described in the literature (D.1, D.20, D.3, D.19 and D.7). General Electric has recently completed development of a microprocessor based system which carries out an algorithm closely related to the first algorithm described in the following paragraphs. The general approach for most existing algorithms consists of the following sequence of three operations.

1. Surface normal vectors are computed for each grid point by using neighboring elevations to approximate local surface slope.

2. These normal vectors are used to generate surface shading at each point by assuming surface brightness is a function of the angle between the normal and an illumination vector.

3. The result, which resembles a shaded relief map, is used to generate a perspective image by ray tracing terrain points through screen pixels to a single viewpoint.

Details follow.

3.2.2 SURFACE NORMAL COMPUTATION

Figure 3.2.2-1 depicts a 30 mile × 30 mile region of the Cascade Mountain Range in northern Washington State. Each of the 512 × 512 pixels represents a single DMA Level I elevation sample point, with elevation represented by pixel brightness. Thus Mount Baker, a high point, appears as the bright mass in the upper left and Lake Shannon, a low area, occupies the dark trough in the middle. This fine resolution grid can be used to determine local earth surface normals for purposes of sun-shading by taking differences between neighboring elevation values as described below.

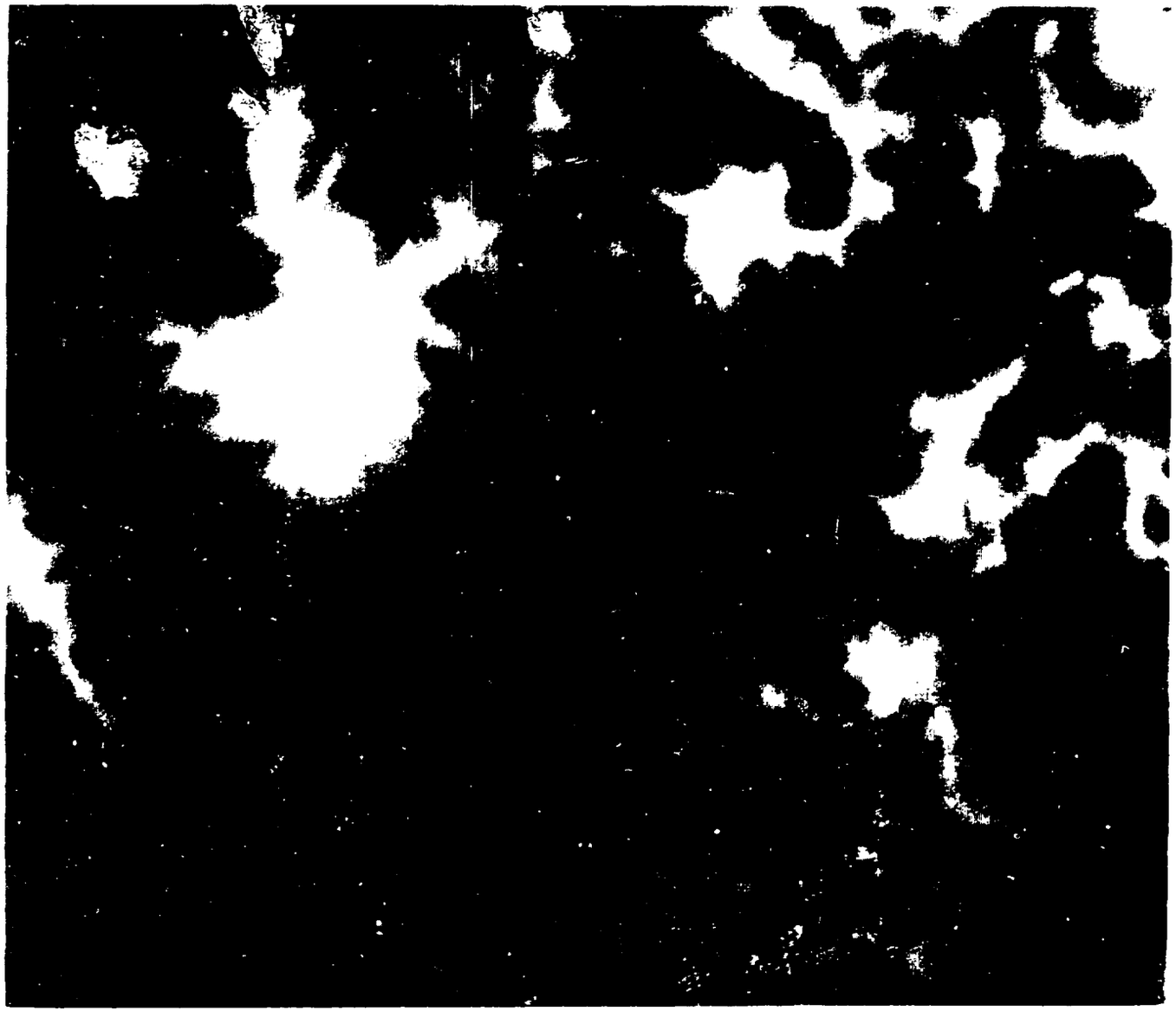


Figure 3.2.2-1. Elevation Displayed as Brightness

Figure 3.2.2-2 represents a microscopic view of a single pixel in Figure 3.2.2-1 and its eight neighbors. The grid spacing Δy is about 300 feet in this particular case. To develop the general method for local surface orientation, however, both Δx and Δy are normalized to unity. Hence (i, j) may be used to denote the space point $(i\Delta x, j\Delta y)$.

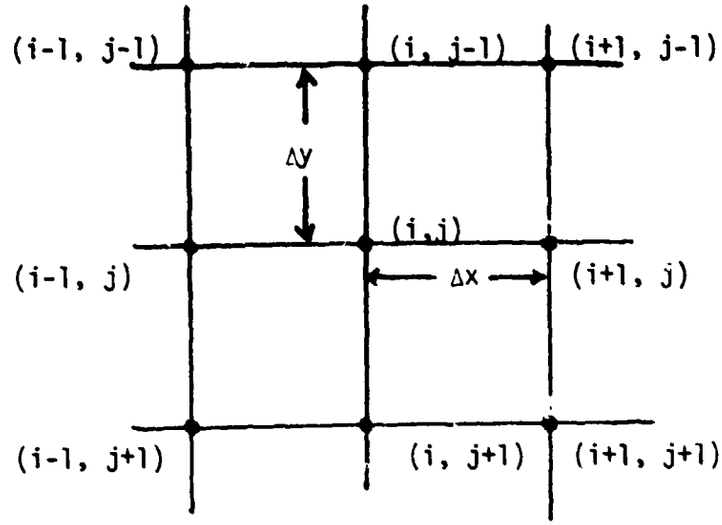


Figure 3.2.2-2. Terrain Grid Notation

The values of H at the two points (x, y) and $(x+\Delta x, y+\Delta y)$ are related by the Taylor expansion.

$$(1)\quad H(x+\Delta x,\ y+\Delta y) = H(x,y) + \left(\Delta x \frac{\partial}{\partial x} + \Delta y \frac{\partial}{\partial y}\right) H(x,y) + \frac{1}{2!}\left(\Delta x \frac{\partial}{\partial x} + \Delta y \frac{\partial}{\partial y}\right)^2 H(x,y) + \cdots + \frac{1}{(n-1)!}\left(\Delta x \frac{\partial}{\partial x} + \Delta y \frac{\partial}{\partial y}\right)^{n-1} H(x,y) + Rn.$$

The remainder term Rn is defined as follows:

$$Rn = \frac{1}{n!}\left(\Delta x \frac{\partial}{\partial x} + \Delta y \frac{\partial}{\partial y}\right)^n H(x + \S\Delta x, y + \S\Delta y) =$$

$$\Theta\left[(|\Delta x| + |\Delta y|)^n\right], \text{ where } o \angle \S \angle 1.$$

Again use the grid point (i, j) to represent the space point (iΔx, jΔy). Expanding H (i-Δx, j) and H (i+Δx, j) about the central value of H (i, j) gives:

$$(2)\ H(i-\Delta x, j) = H(i, j) - \Delta x \frac{\partial H(i, j)}{\partial x} + \frac{(\Delta x)^2}{2!} \frac{\partial^2 H(i, j)}{\partial x^2}$$
$$- \frac{(\Delta x)^3}{3!} \frac{\partial^3 H(i, j)}{\partial x^3} + \frac{(\Delta x)^4}{4!} \frac{\partial^4 H(i, j)}{\partial x^4} + \ldots$$

$$(3)\ H(i + \Delta x, j) = H(i, j) + \frac{\Delta x \partial H(i, j)}{\partial x} + \frac{(\Delta x)^2}{2!} \frac{\partial^2 H(i,j)}{\partial x^2}$$
$$+ \frac{(\Delta x)^3}{3!} \frac{\partial^3 H(i,j)}{\partial x^3} + \frac{(\Delta x)^4}{4!} \frac{\partial^4 H(i,j)}{\partial x^4} + \ldots$$

Analogously,

$$(4)\ H(i, j + \Delta y) = H(i, j) + \Delta y \frac{\partial H(i, j)}{\partial y} + \frac{(\Delta y)^2}{2!} \frac{\partial^2 H(i, j)}{\partial y^2}$$
$$+ \frac{(\Delta y)^3}{3!} \frac{\partial^3 H(i,j)}{\partial y^3} + \frac{(\Delta y)^4}{4!} \frac{\partial^4 H(i,j)}{\partial y^4} + \ldots$$

$$(5)\ H(i,j - \Delta y) = H(i,j) - \frac{\Delta y \partial H(i,j,)}{\partial y} + \frac{(\Delta y)^2}{2!} \frac{\partial^2 H(i,j)}{\partial y^2}$$
$$- \frac{(\Delta y)^3}{3!} \frac{\partial^3 H(i,j)}{\partial y^3} + \frac{(\Delta y)^4}{4!} \frac{\partial^4 H(i,j)}{\partial y^4} + \ldots$$

Now letting $\Delta x = \Delta y = 1$, we can solve equations (2) through (5) to obtain the derivatives in the X and Y directions at point (i, j).

(6) Backward x- Difference

$$\frac{\partial H(i,j)}{\partial x} = \frac{H(i,j) - H(i-1,j)}{\Delta x} + \mathcal{O}(\Delta x) \simeq H(i,j) - H(i-1,j)$$

(7) Forward x- Difference

$$\frac{\partial H(i,j)}{\partial x} = \frac{H(i+1,j) - H(i,j)}{\Delta x} + \mathcal{O}(\Delta x) \simeq H(i+1,j) - H(i,j)$$

(8) Central x- Difference

$$\frac{\partial H(i,j)}{\partial x} = \frac{H(i+1,j) - H(i-1,j)}{2\Delta x} + \mathcal{O}(\Delta x)^2 \simeq H(i+1,j) - H(i-1,j)$$

(9) Backward y- Difference

$$\frac{\partial H(i,j)}{\partial y} \simeq H(i,j) - H(i,j-1)$$

(10) Forward y- Difference

$$\frac{\partial H(i,j)}{\partial y} \simeq H(i,j+1) - H(i,j)$$

(11) Central y- Difference

$$\frac{\partial H(i,j)}{\partial y} \simeq H(i,j+1) - H(i,j-1)$$

The unit surface normal at grid point (i, j) is given by:

(12) $$\vec{n}(x,y) = \left[-\frac{\partial H(i,j)}{C\partial x} , -\frac{H(i,j)}{C\partial y} , \frac{1}{C} \right]^t$$

Where $$C = \left[1 + \left(\frac{\partial H(i,j)}{\partial x} \right)^2 + \left(\frac{\partial H(i,j)}{\partial y} \right)^2 \right]^{\frac{1}{2}}.$$

In order to prevent the derivative field from being offset by one-half a grid point distance in the X and Y directions, it is probably preferable to use the central differences when computing $\vec{n}(x,y)$.

3.2.3 SURFACE SHADING COMPUTATION

Assume that a height field H(x, y) is illuminated by a point light source at infinity and ambient light. The location of this point light source can be given in terms of its azimuth α and elevation Ω (see Figure 3.2.3-1). Under the convention used for sunlight, azimuth is measured clockwise from North, while elevation is the angle between the light source and the horizon.

Some of the light reaching a surface is reflected, while the remainder passes on into the substance of the body to be absorbed. The reflected light has components which may be characterized as either diffuse or specular. A point on a purely diffusely reflecting surface has the same brightness at every angle of observation. Differences in brightness result from differences in the amount of incident light per unit area intercepted by portions of the surface at various angles to the light source. According to Lambert's Law of Diffuse Reflection, the amount of light reaching an observer from a surface element is related to the intensity of the light source P_λ for frequency λ and the cosine of the incident angle i (Figure 3.2.3-2). A negative cosine value indicates that the light source is on the opposite side of the object from the surface element. If the object is opaque, the contribution from this surface element is zero.

Let $\vec{n}(x, y)$ be the unit surface normal at point (x, y) and $\vec{n}_p$ be a unit vector in the direction of the light source. Then, the diffuse term can be expressed as

$$\max\left[0, P_\lambda \vec{n}(x,y) \cdot \vec{n}_p\right] = \begin{cases} P_\lambda \cos i & \text{for } |i| < \pi/2 \\ 0, \text{ otherwise} \end{cases} \qquad (3.2.3\text{-}1)$$

where

$$\vec{n}(x,y) = \left[\frac{-\alpha H(x,y)}{C\alpha x}, \frac{-\alpha H(x,y)}{C\alpha y}, 1/C\right]^T,$$

$$C = \left[1 + \left(\frac{\alpha H(x,y)}{\alpha x}\right)^2 + \left(\frac{\alpha H(x,y)}{\alpha y}\right)^2\right]^{1/2},$$

and

$$\vec{n}_p = (\sin\alpha \cos\Omega, \cos\alpha \cos\Omega, \sin\Omega)^T$$

A specular reflector is mirror-like; the angle of reflection of most of the light is equal to the angle of incidence. For a perfect mirror surface, light would only reach an observer if the surface normal fell half way between the source direction and the

Ω = Angular Elevation of the Sun;

α = Sun Azimuth

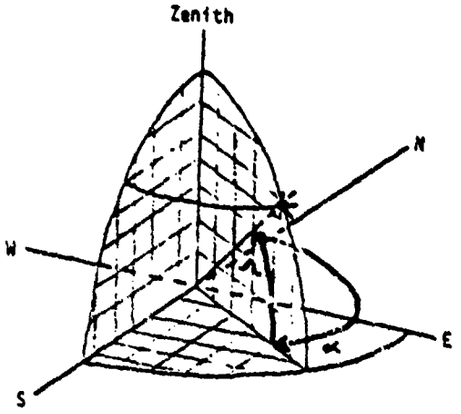
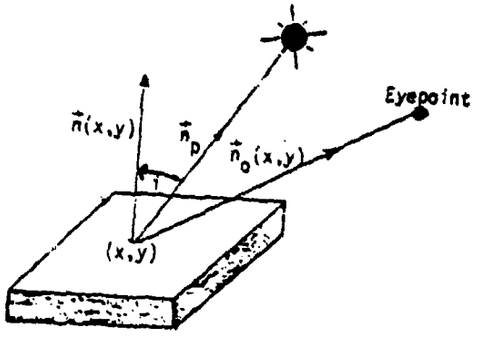


Figure 3.2.3-1. Height Field Illumination by Point Source



1EA525

Figure 3.2.3-2. Unit Direction Vectors

eye direction. For less than perfect reflectors, the amount of specular reflection reaching the eye is related to a power of the cosine of the angle between the direction of reflected light and the viewline. The specular term can be expressed as

$$(\vec{n}(x,y)\cdot\vec{n}_p)(\vec{n}_r(x,y)\cdot\vec{n}_o(x,y))^k P_\lambda \text{ for } (\vec{n}(x,y)\cdot\vec{n}_p) \text{ and } (\vec{n}_r(x,y)\cdot\vec{n}_o(x,y))>0 \qquad (3.2.3\text{-}2)$$

where

$$(\vec{n}_r(x,y)\cdot\vec{n}_o(x,y)) = 2(\vec{n}(x,y)\cdot\vec{n}_p)(\vec{n}(x,y)\cdot\vec{n}_o(x,y)) - (\vec{n}_p\cdot\vec{n}_o(x,y)),$$

where $\vec{n}_r(x,y)$ is the unit vector at point (x,y) in the direction of the reflected light, $\vec{n}_o(x,y)$ is the unit vector at point (x,y) in the direction of the observer, and k is dependent on the shininess of the surface. A high value of k corresponds to a small surface glare point (highlight).

Ambient light strikes a surface point equally from all directions and thus is reflected equally in all directions. The intensity of ambient light for wavelength λ will be denoted by A_λ.

The albedo of an object is the fraction of the incident energy which is reflected by the object over the entire electromagnetic bandwidth. The albedo of a surface for frequency λ will be denoted by r_λ. Tables exist for r_λ for many natural materials (D.15, D.16, D.17). These numbers are useful for simulating realistic images of natural materials. The visual albedo of an object refers to the object's albedo as perceived by the human visual system. It is the product of the object's albedo and the spectral luminosity of the human eye.

A number of simplifying assumptions are required in order to present an equation for the appearance of a surface. Assume that light reflected from one surface element does not illuminate another surface element, since such an effect is difficult to model and will lead to global computations. In addition, all surfaces will be assumed to be nonfluorescent; i.e., they are not sources of light. Also assume that surface elements do not cast shadows on each other. Under the above assumptions and simplifications, the appearance of surface point (x,y) over the visible bandwidth is given by

$$\begin{aligned} R(x,y) = {} & \kappa \max\{0\ [\vec{n}(x,y)\ \vec{n}_p]\}\, r_\lambda(x,y)P_\lambda \\ & + (1-\kappa)\max\{0, [\vec{n}(x,y)\cdot\vec{n}_p]\}(\max\{0, [\vec{n}_r(x,y)\cdot n_o(x,y)]\})^k r_\lambda(x,y)P_\lambda \\ & + r_\lambda(x,y)A_\lambda, \quad \text{for } 380\text{nm} < \lambda < 770\text{nm}, \end{aligned} \qquad (3.2.3\text{-}3)$$

where κ is the proportion of light from the point light source which is diffusely reflected. The three terms correspond respectively to diffuse, specular and ambient components.

Before an image is displayed on a color cathode-ray tube, frequency values are usually converted first to CIE tristimulus values and then to red, green, and blue intensity levels.

For many applications, equation (3.2.3-3) is unnecessarily complex. A simple monochromatic model for the appearance of a height field can be obtained by eliminating the specular reflectance term and making all other terms independent of wavelength. Under these simplifications, equation 3.2.3-3 becomes

$$R(x,y) = P \max \{0, [\vec{n}(x,y) \cdot \vec{n}_p]\} + A. \qquad (3.2.3\text{-}4)$$

Figure 3.2.3-3 illustrates the application of shading to the data displayed in Figure 3.2.2-1. Note how much more vividly terrain relief stands out than in the elevation display. Figure 3.2.3-4 illustrates the DMA planimetric surface material color of this region, modulated by the sun shading values. This image is the source data for the perspective images discussed in the following paragraph.

3.2.4 PERSPECTIVE DISPLAY COMPUTATION

Perspective display of data of the type shown in the preceding paragraphs is generated by tracing rays from a viewpoint through screen pixels to the first terrain elevation points encountered in the grid data. A pixel is assigned the color of the shaded culture image (Figure 3.2.3-4) found at that grid location. There are many possible ways to sequence through the grid data in order to optimize memory access and processing. Each attempts to take advantage of data or display coherence to avoid duplicating computations. Data access order and computational complexity are crucial factors in the task of efficiently carrying out the enormous number of steps characteristic of grid display. The drawback of computation volume may be more than offset by the simplicity of the individual steps embodied in efficient compact hardware. For example, under an IR&D program, General Electric has developed a microprocessor based Mission Planning and Briefing Console which implements such algorithms.

The following paragraphs describe two different methods of processing grid data for perspective display. The first is based on efficient sequential algorithms and the second is based on new parallel algorithms.

Figure 3.2.3-3. Sun Shading of Elevation Data

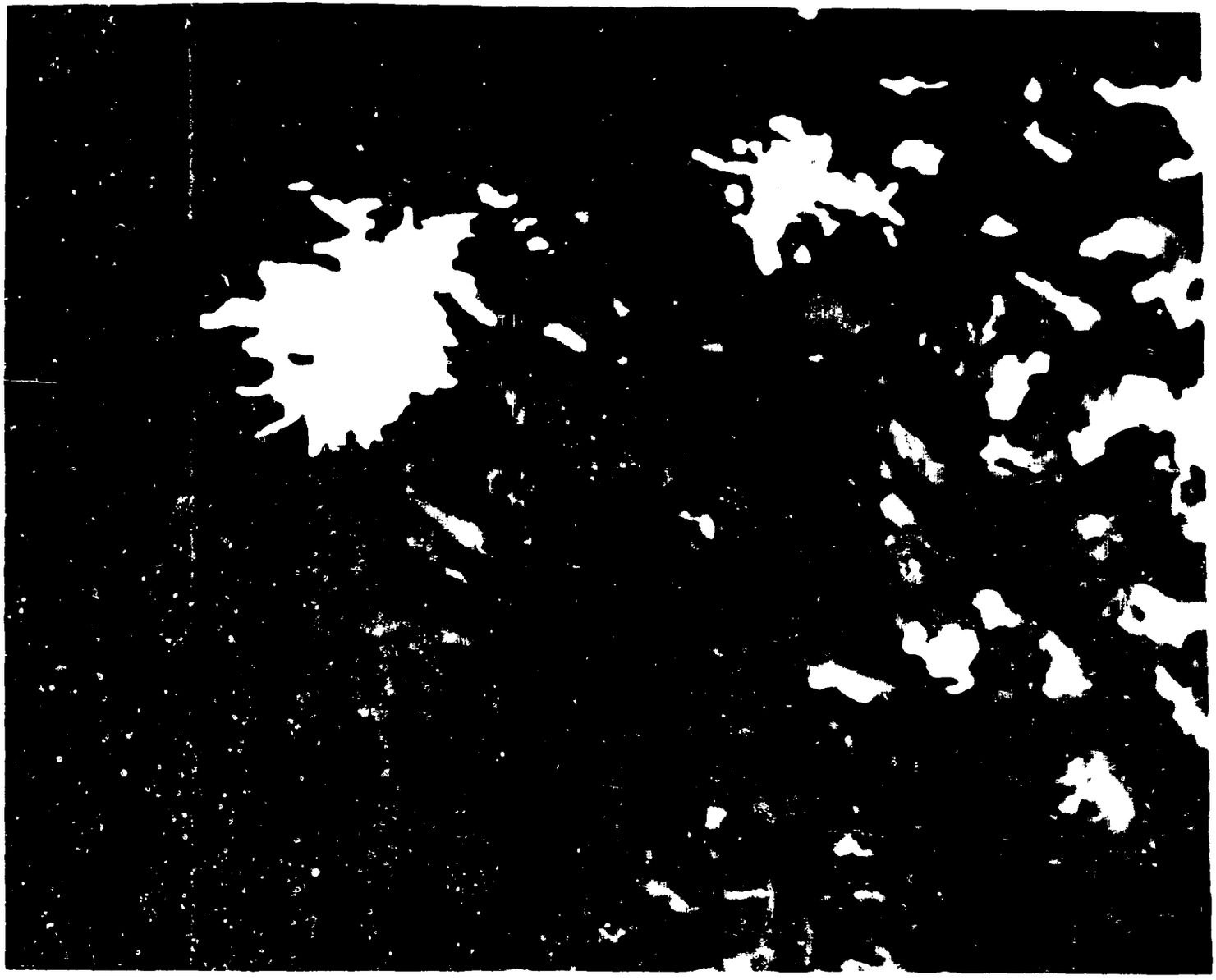


Figure 3.2.3-4. Combined Sun-Shaded and Culture-Colored Terrain

3.2.4.1 Azimuth Ray Tracing Algorithm

Figure 3.2.4.1-1 illustrates the basic geometry of the projection of grid data onto the viewscreen. Each pixel is projected onto a sample point of the grid. The color associated with that point is then assigned to the pixel. The order in which data is scanned is illustrated in Figure 3.2.4.1-2. Basically, the screen is divided into columns of pixels (c) which correspond to equal azimuth sectors (a) cutting through the data base. Each screen column is swept from bottom to top so that screen raster increments correspond to successive range increments in the data base (b). The final result is an image such as shown in Figure 3.2.4.1-3. Various enhancements may be introduced such as hazing or texturing. Distance hazing caused by atmospheric scattering is modeled by blending terrain color with a blue-white haze color as a function of range, namely,

HAZED COLOR = COLOR * (1 -(EXP (-Ki)) * HAZE)

This exponential haze fading is borne out by atmospheric models (C.17, C.18). Figure 3.2.4.1-4 illustrates such an enhanced image.

Azimuth ray tracing algorithms generate undesirable artifacts at very long ranges and very short ranges due to oversampling and undersampling of the data grid by pixel projections. However, excellent images have been generated by applying oversampling and smoothing techniques. Figure 3.2.4.1-3 was generated by this technique. The latter is a serious problem for low level sensor imagery. Another drawback of azimuth ray tracing is the random accessing of grid data memory which prevents efficient parallel processing. The algorithm described in the next paragraph avoids such problems.

3.2.4.2 Parallel Computation Algorithm

During the investigation and enhancement of existing grid data base display algorithms, a new algorithm was devised which is radically different from existing methods. Although the algorithm has not yet been implemented, the design reveals some promising new properties which suggest highly efficient hardware or software implementation.

The new algorithm accesses rows and columns of data independently. It completely decouples depth related calculations (haze, occulting, and perspective) from scanline related calculations (azimuth ray images). Such properties encourage parallelism in the design of compact, inexpensive hardware systems for generating realistic, geometrically valid perspective images of terrain directly from Defense Mapping Agency (DMA) data. These properties also provide for straightforward

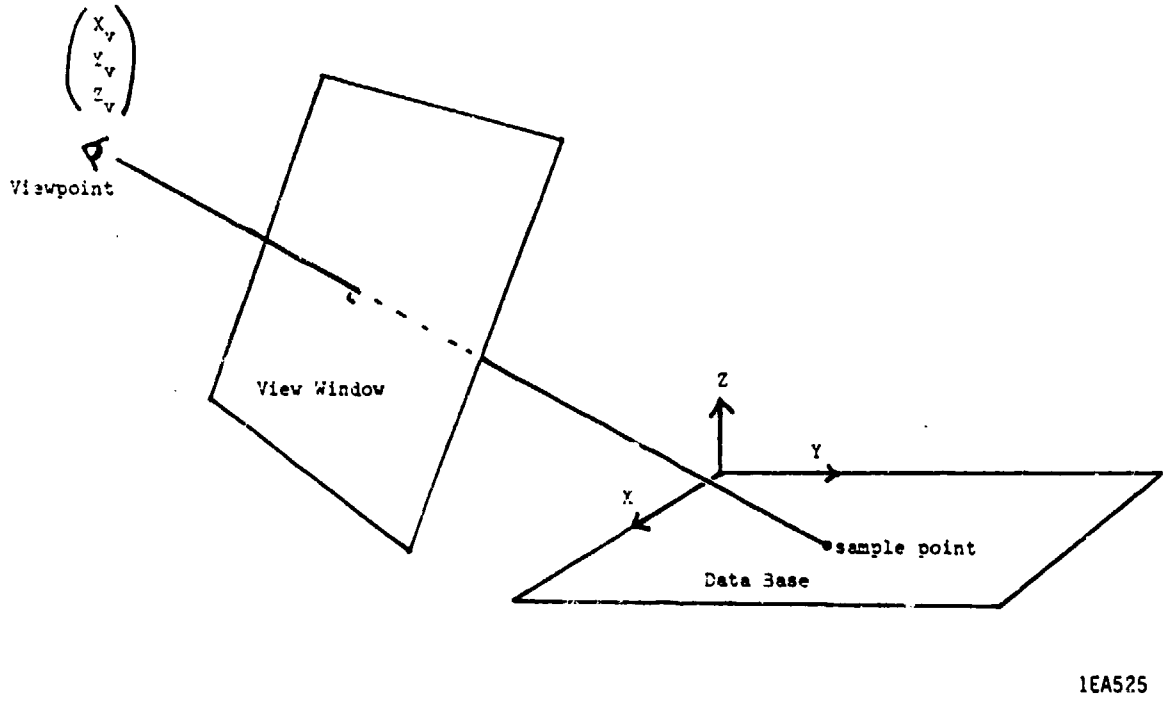


Figure 3.2.4.1-1. Viewpoint, View Window, and Data Base Relationships

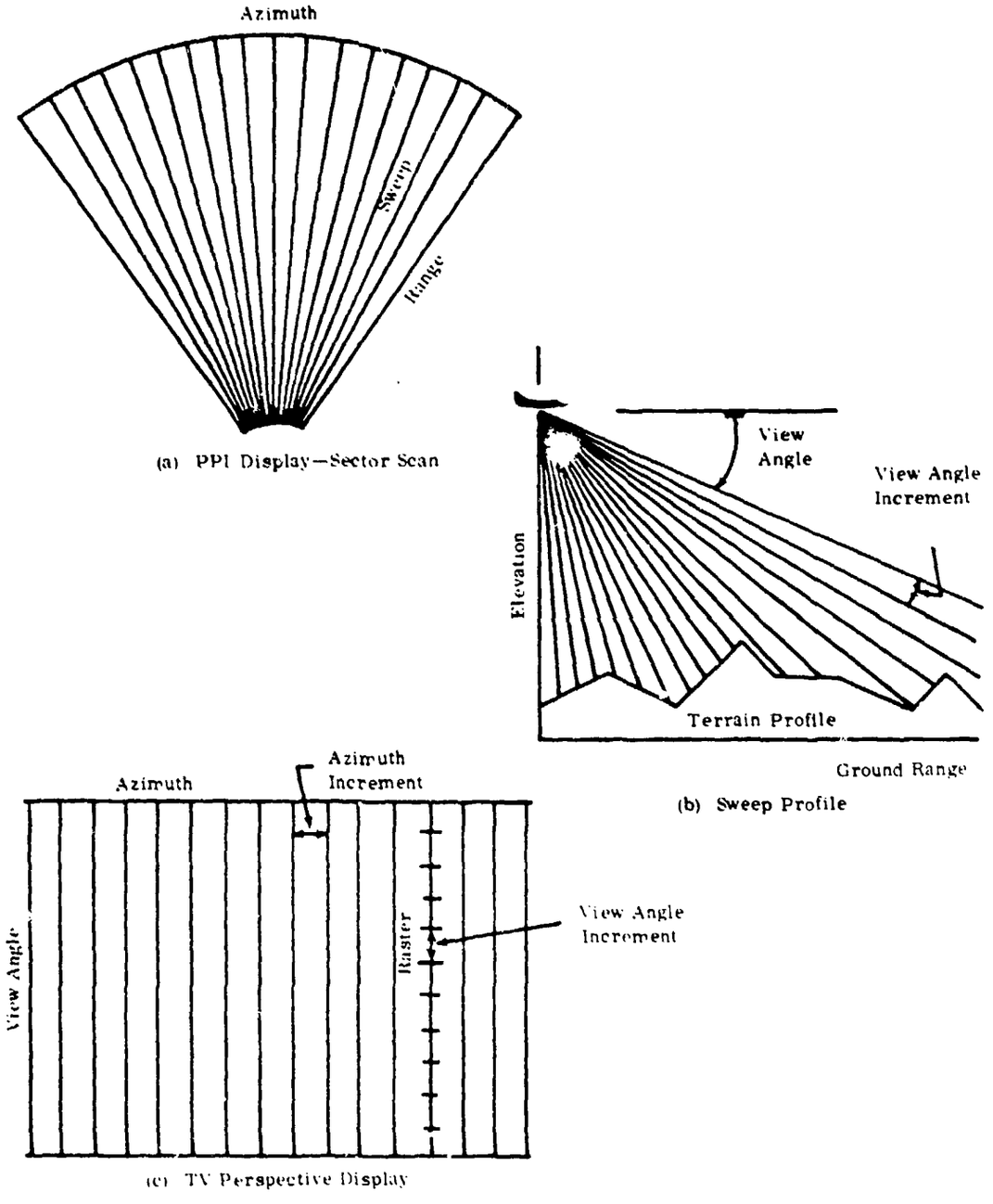


Figure 3.2.4.1-2. Grid Data Projection Onto Viewscreen

Figure 3.2.4.1-3. Perspective Display of Grid Data

Figure 3.2.4.1-4. Enhanced Perspective Display of Grid Data

enhancement of close range scenes by elevation interpolation and the interleaving of models or special features during either data definition or display generation. The grid data structure is ideal for fractal interpolation to generate realistic terrain images at close ranges without storing the corresponding elevation data.

The algorithm is designed to render perspectively correct color displays of grid data bases. Anti-aliasing, valid occlusion of invisible features (priority) and distance fading (haze) calculations are inherent in the algorithm. Computational organization exploits highly parallel hardware efficiency by making row and column calculations simple, local and independent. Trigonometric functions are avoided entirely and access to the grid data base is by row-column rather than random. There are five major steps to the algorithm, each of which incorporates row or column independent parallel computation, described below. The grid data base is assumed to contain color and elevation data just as in ray tracing algorithms.

Step 1: Rotate the grid data base by the negative of the yaw angle of viewer attitude. This aligns the boresight with the data base axis, a crucial step in achieving row and column independence in subsequent steps. The grid rotation can be carried out by row and column parallel operations as described in (D. 21).

Step 2: Perform a perspective transformation on the elevation grid by independent parallel row operations. This is basically a resampling process which yields elevation points at fixed distance from the viewplane, lined up along constant azimuth planes. It is carried out in parallel by grid row.

Step 3: Trace the perspectively transformed azimuth grid outward, calculating the color contribution of successive elevation points to successive pixels including such effects as distance haze fading, anti-aliased color transition, and valid occlusion of invisible data.

Step 4: Correct the perspective distortion associated with viewplane pitch by applying a parallel transformation to screen data by computations resembling those of Step 2. That is, screen rows are compressed according to pixel footprint width, but this time there are no elevation calculations, only color calculation.

Step 5: Rotate screen data to account for view attitude roll in the same manner as the first step, but on screen color data rather than the data base.

Obviously Steps 4 and 5 could be omitted entirely if the roll and pitch of the viewplane are both zero. Details of these steps are described below in paragraphs 3.2.4.2.1 through 3.2.4.2.5.

3.2.4.2.1 Step 1

The efficiency of the entire algorithm depends on the fact that the grid coordinate system is aligned with the viewray through the viewpoint perpendicular to the viewscreen. As can be seen in Figure 3.2.4.2-8, this results in constant range for all grid points in a row parallel to the screen, which greatly simplifies data access and computation.

Grid rotation by matrix multiplication of point coordinates is computationally expensive, slow and complicated in terms of order of data access. Hence we use a much simpler, faster, and more parallel method described in detail in (D.21). Briefly, the method of rotation consists in decomposing the transformation into four simpler ones, two shears, a shrink, and a stretch as illustrated in Figure 3.2.4.2-1. Each of these simple transformations moves data parallel to one of the grid axes. Geometrically valid accounting for the discrete nature of grid data is achieved by computing new data values as sums of old values weighted by the area of overlap of grid cells as illustrated in Figure 3.2.4.2-2. The weighting coefficients may be calculated efficiently from the slope of the shear by modulo arithmetic carried out by the processor illustrated in Figure 3.2.4.2-3. The derivation of this calculation is based on a discrete code for linear transformations invented by Rothstein described in (G.29).

While the derivation is too lengthy for inclusion here, the result is that each weighting coefficient can be calculated by a single fixed point addition of two numbers whose precision is that of the grid; i.e., 10 bits for a 1024×1024 grid.

The modulo arithmetic unit can be used to control source and destination grid addresses and generate the coefficients to drive the digital filter which routes data through the transformation as illustrated in Figure 3.2.4.2-4. Figure 3.2.4.2-5 illustrates the sequence of data transformations and shows that they may be carried out in parallel. With a 1 MHz instruction cycle time, a complete 1000×1000 point grid can be rotated in four milliseconds, almost an order of magnitude faster than required by video frame rates. Figure 3.2.4.2-6 illustrates a software simulation of the algorithm on a grid image. Note the total absence of aliasing effects and the preservation of detail. This implies that rotating a terrain grid by this method would not introduce any instability to elevation; i.e., the ground would not ripple as view attitude changes in real time.

3.2.4.2.2 Step 2

Motivation for the perspective transformation of grid data can best be understood by looking at Figures 3.2.4.2-7 and 3.2.4.2-8, which represent side and top views of the screen projection configuration. Figure 3.2.4.2-7 is a vertical slice through the

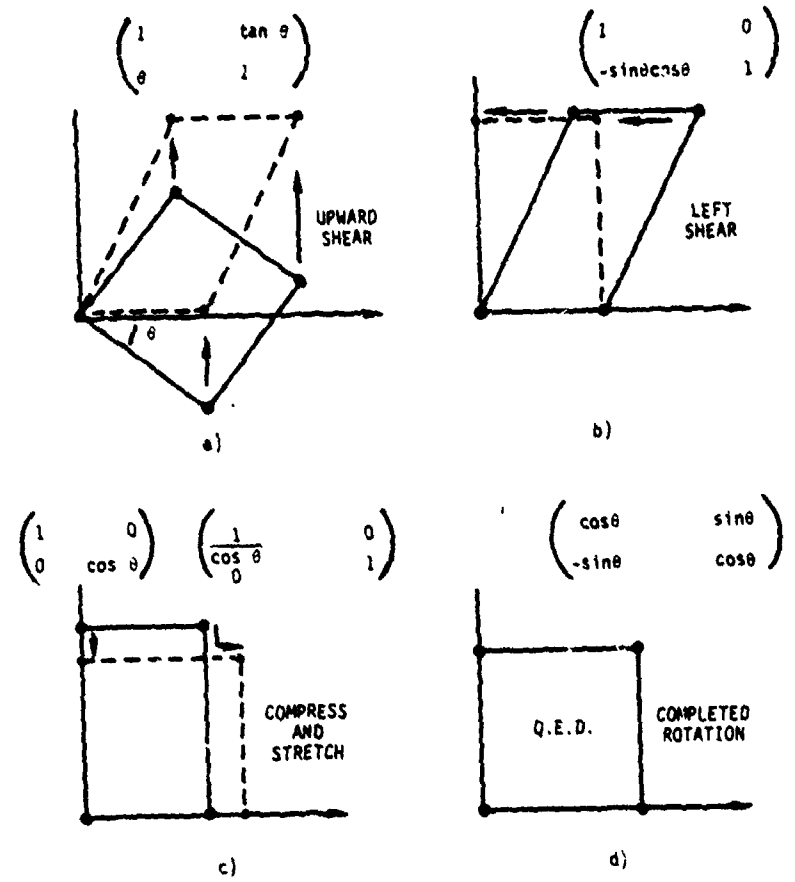


Figure 3.2.4.2-1. Decomposition of Rotation into Axis-Parallel Shearing and Scaling

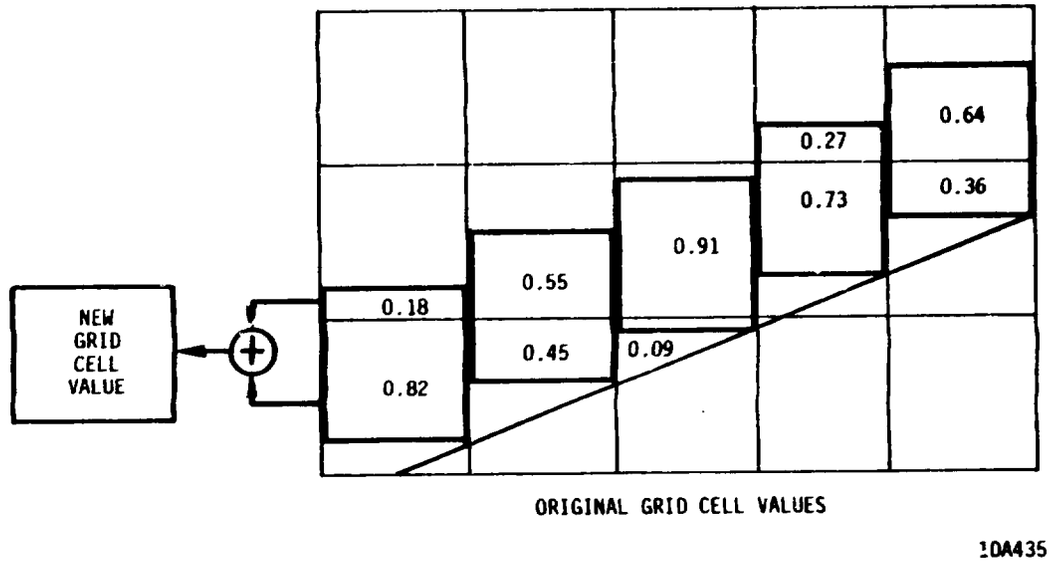


Figure 3.2.4.2-2. Shearing as a Shifting Weighted Sum of Overlapped Grid Cells

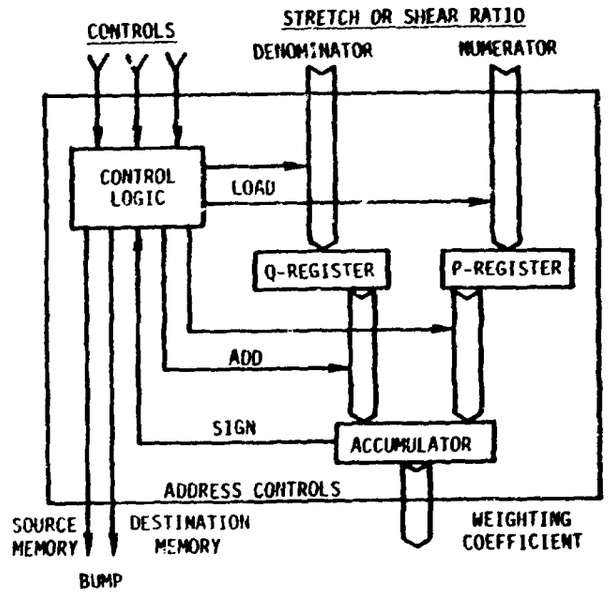


Figure 3.2.4.2-3. Modulo Arithmetic Unit

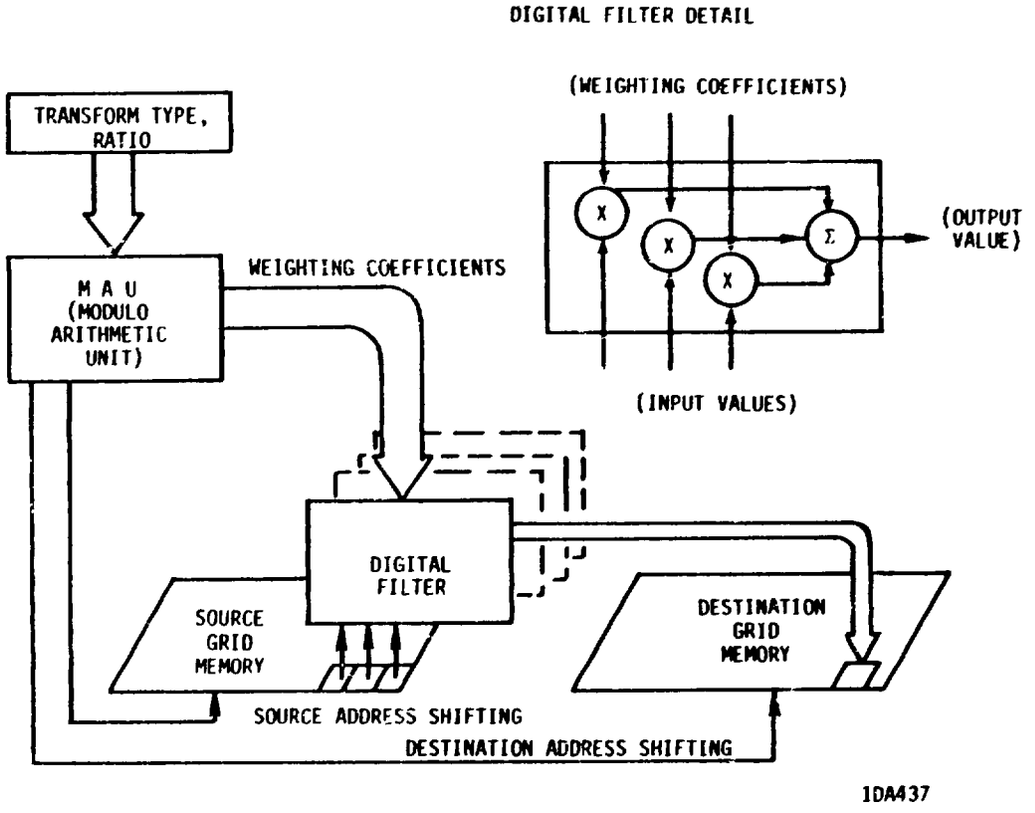
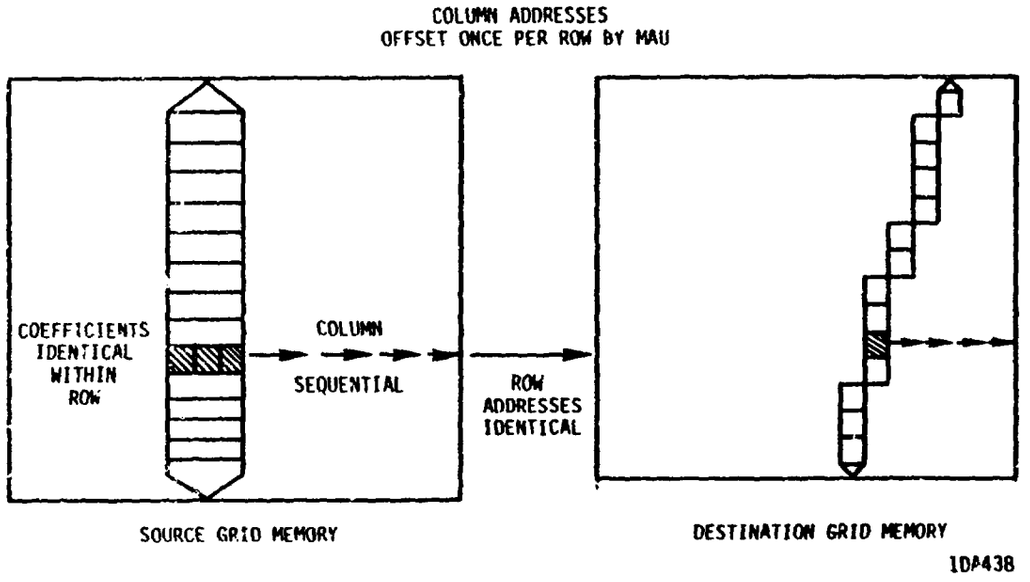


Figure 3.2.4.2-4. Grid Rotation System Architecture



Figur 3.2.4.2-5. Computation Structure for Horizontal Shearing

Figure 3.2.4.2-6. Rothstein Code Rotation by 30 Degrees

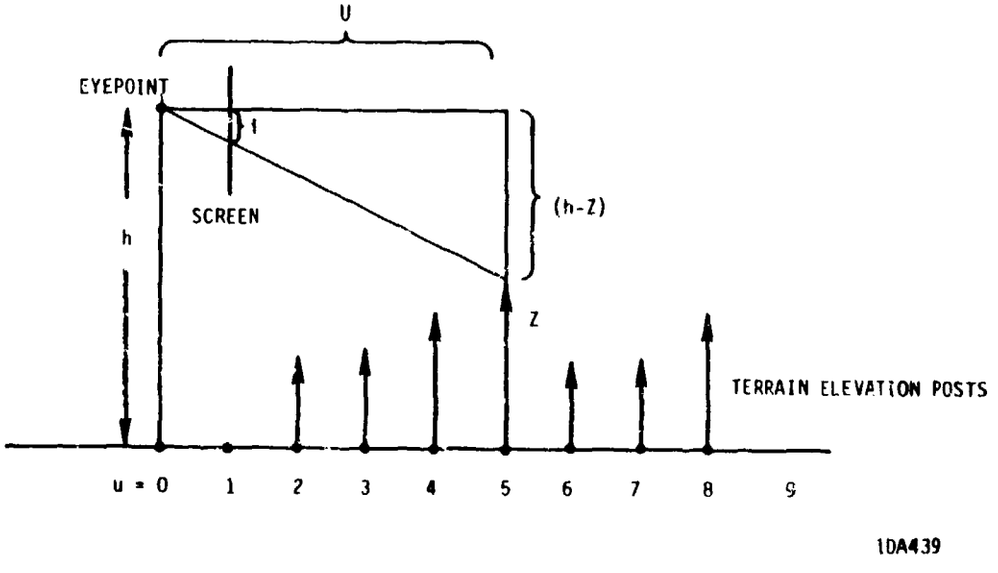


Figure 3.2.4.2-7. Side View of Perspective Screen Projection

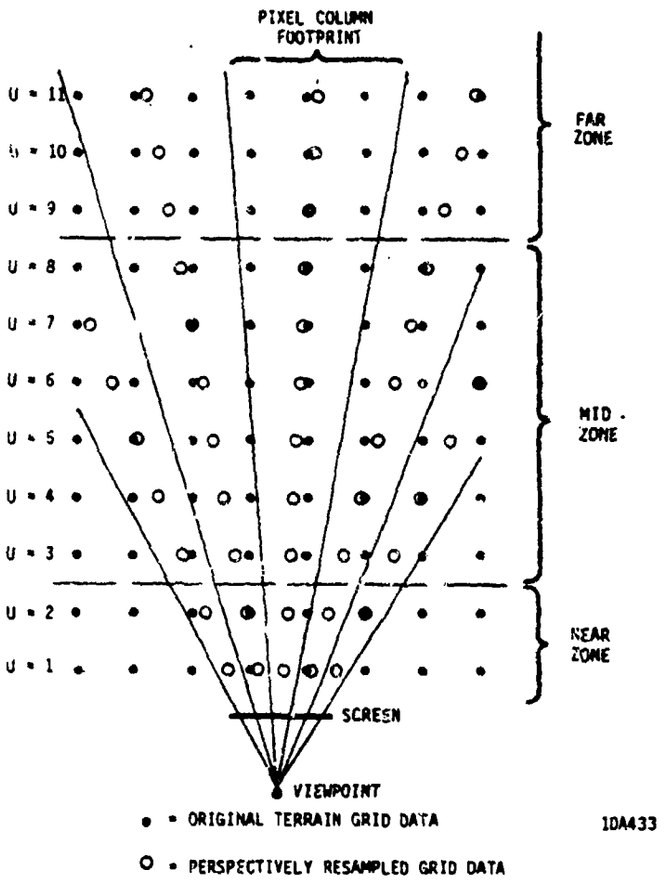


Figure 3.2.4.2-8. Top View of Perspective Resampling of Grid Data

viewpoint. Refer to the terrain grid coordinates as U (away from the viewpoint) and V (to the right horizontally parallel to the screen). A line of grid points at constant U value is referred to as a row, and a line of grid points of constant V is referred to as a column of data. Associated with each grid point is a terrain elevation Z(U, V) above the zero level, and a specification of the color C(U, V) at that point. If the eyepoint is at height h above the zero elevation plane, then the data at (U, V) will project to the screen at pixel position.

$$\text{row } i = \frac{h-Z}{U}, \quad \text{column } j = \frac{V}{U}.$$

The color of that pixel should be C(U, V). This straightforward approach leads to some geometric problems which are avoided by the proposed algorithm.

At long ranges, several elevation posts contribute to the image at one pixel. At close ranges, the opposite is true; each elevation post contributes to several pixels. A straight viewray algorithm thus undersamples distant points and oversamples close ones. Visual artifacts for distant views consist of terrain elevation discontinuities which change for changing yaw angles. At close ranges, the effect is a blocky appearance of terrain with abrupt elevation changes between elevation points. Both visual artifacts can be avoided by resampling the data rows to generate new elevation posts whose spacing is precisely that of viewrays through laterally adjacent pixels as shown by the small circles in Figure 3.2.4.2-8. Note that because the view screen is aligned with data base coordinates, the viewray intersections with a row of elevation posts are equally spaced at any particular range. This property permits the use of a fixed resampling scheme for all points in any row of constant range. The computational advantages are that data may be accessed in stream fashion rather than randomly, and the resampling parameters need only be updated once per data row.

Figure 3.2.4.2-9 illustrates a variety of near zone interpolation schemes; the dark dots correspond to grid data points at constant U values (in the same row). The pixel sample points would fall between these grid points. Elevation values are to be assigned as values of the interpolating curve. The zero(th) order case corresponds to oversampling characterized by straight ray tracing methods and yields blocky looking close terrain. First order interpolation eliminates the discontinuities between the blocks. Higher order interpolation yields surfaces with higher degrees of continuity.

The interpolation scheme in the near zone may be quite arbitrarily chosen to satisfy a wide variety of criteria. While smoothness is often desired in interpolation of functions, it may be quite undesirable in simulation because roughness is a much

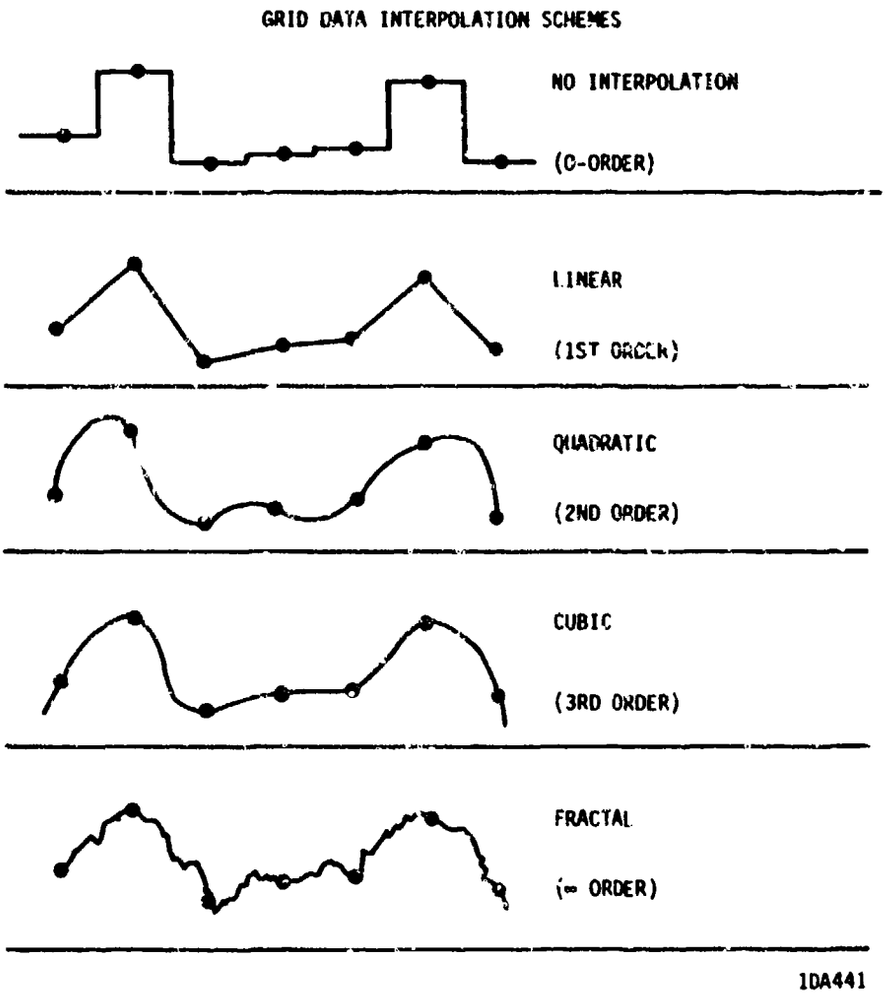


Figure 3.2.4.2-9. Alternative Interpolation Algorithms for Near Zone (Large dots represent data base grid points.)

better visual cue. Recently in computer graphics (G.5), random processes have been used to generate strikingly realistic artificial terrain. These processes yield surfaces and curves called fractals; they have many of the shape and roughness properties of natural terrain. Though their mathematical expression is rather complex, computed values of fractals can be stored in a lookup table. Such a lookup table can be used for interpolation between data points with less computation than any of the polynomial methods.

Figure 3.2.4.2-10 illustrates a fractal property of statistical self-similarity. Though the sampling rates of the superimposed curves differ by almost an order of magnitude, they match position very closely. This crucial property of fractals assures unlimited continuous level-of-detail change to microscopic levels using a single lookup table, while preserving visual stability of the synthesized terrain.

Fractal methods are particularly suitable for grid data bases because of the geometric coherence (geometric neighbors are stored as memory neighbors) of the data structure. In polygonal network data bases, new levels of detail require multilevel pointer indexing through randomly stored data. This requires searching and sorting, computations whose complexity increases worse than linearly with data quantity.

The resampling schemes in all three zones generate horizontal perspective corrected grid data by computations involving multiplication and addition. By properly scaling the multiplication, division by distance from viewpoint can be carried out at the same time to yield vertical, perspective corrected elevation. That is, all elevations in a grid row can be divided by the constant U characterizing that row during the resampling process. The result of this operation on the data shown in Figure 3.2.4.2-7 is shown in Figure 3.2.4.2-11. The corrected elevations may be used directly for priority (occlusion of visibility) as described in the following section.

3.2.4.2.3 Step 3

Now that the ground positions of grid points have been perspectively corrected in Step 2, each column of data projects to a single column of pixels on the screen. Hence, occlusion of visibility for any terrain data can only be caused by data in the same column with a smaller value of U; i.e., closer to the viewpoint. This geometric property provides a very simple, parallel priority rule which requires no sorting whatsoever.

Let Figure 3.2.4.2-11 represent a side view of any column of data and its screen projection. Since elevations have been perspectively corrected, the projection onto the screen is horizontal. Thus, pixel color assignment with correct priority is accomplished simply by iterating upward along a column of pixels without backtracking. At each step to a new pixel, the terrain color is assigned to that pixel.

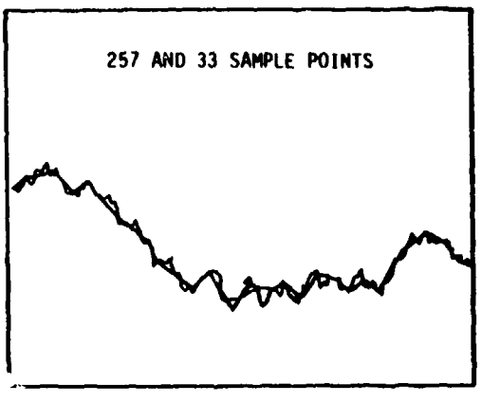


Figure 3.2.4.2-10. Fractal Curves From [Fournier and Fussel, 1980]

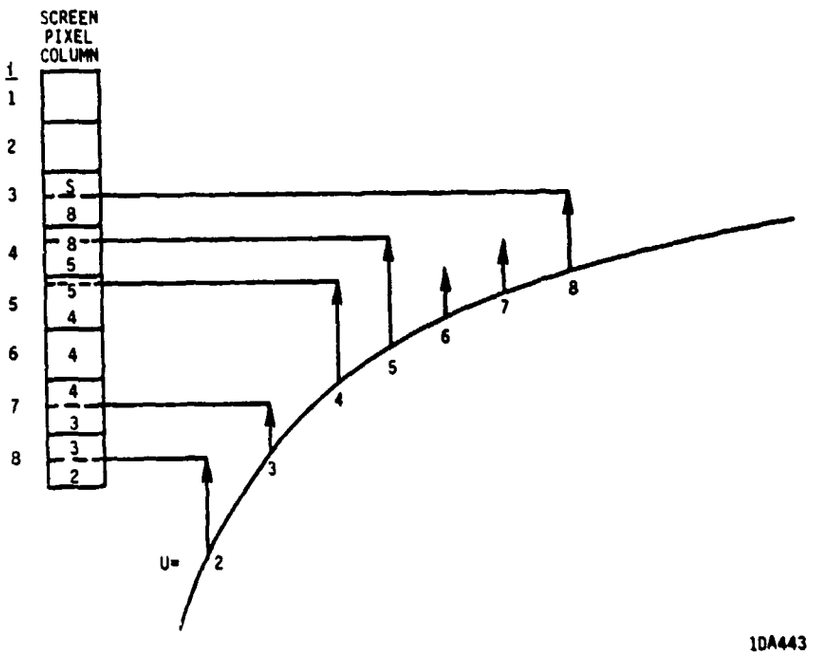


Figure 3.2.4.2-11. Side View of Perspectively Corrected Elevation Posts Projected Onto Screen

Otherwise, the terrain color contributes to the pixel, but weighted by the relative elevation achieved within the pixel. The left column in Figure 3.2.4.2-11 illustrates this process in pixel 7 which receives color contributions from terrain posts 3 and 4. Note in pixel 4 that elevation posts 6 and 7 are hidden behind post 5 and hence contribute nothing. This illustrates the simplicity of implementing priority. When the end of the terrain is reached, remaining pixels are colored with the sky color, denoted by S in Figure 3.2.4.2-11.

Since the U values represent distance from the viewpoint, hazing may be introduced directly into the pixel color assignment process by blending the elevation based assignment of color with haze color using the current value of U to determine weighting coefficients. The value of U may also be used to merge polygonal data base models such as buildings or vehicles. That is, range to terrain is automatically available and can be used to implement any range based priority scheme for models, regardless of their source.

3.2.4.2.4 Step 4

Step 1 corrected view orientation for aircraft yaw by rotating the grid data base to align with direction of view. Step 4 corrects for aircraft pitch by perspectively distorting the image generated in Step 3 as follows.

The vertical screen at the top of Figure 3.2.4.2-12 holds an image generated by Steps 1 through 3. Had the projection originally been to the pitched screen, the ray from any object in the 3-D terrain to the viewpoint is the same ray which generated the corresponding image in the vertical screen. That is, the pitched screen image can be computed directly as a plane projection of the image on the vertical screen, without the need to refer to the 3-D terrain data. The pitched screen image of vertical screen data is illustrated at the bottom of Figure 3.2.4.2-12. Note that pixel rows are parallel in both images; hence the calculation for projection is precisely that used to resample the grid in Step 2, illustrated in Figure 3.2.4.2-8. Rather than resampling terrain data, pixel contents are resampled in Step 4.

3.2.4.2.5 Step 5

The only view attitude orientation remaining uncorrected is aircraft roll. This may be regarded as a rotation about a perpendicular to the viewplane, through the viewpoint. The visual effect of roll may be accounted for by rotating the image depicted at the bottom of Figure 3.2.4.2-12 about the center of the screen. This rotation can be carried out on pixel data exactly as the computation described in Step 1.

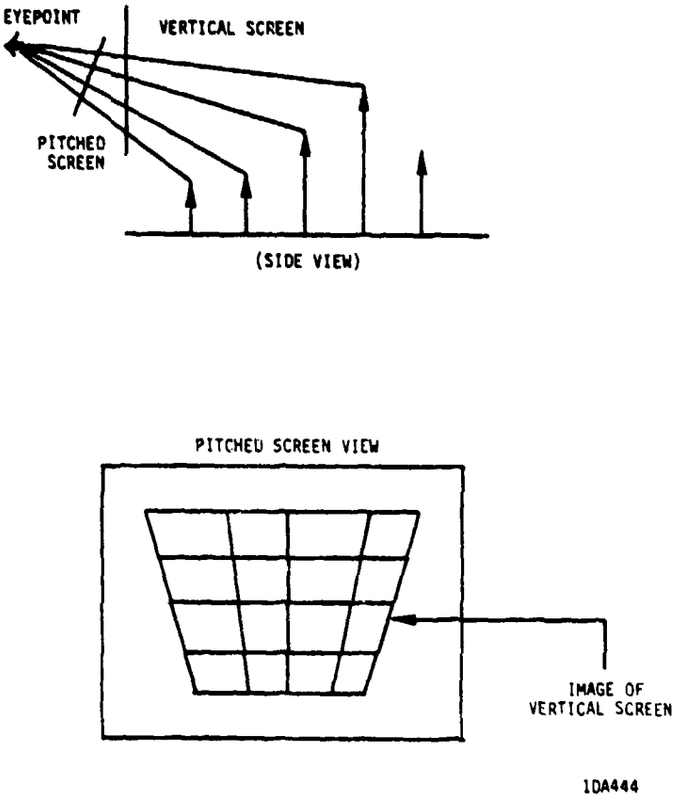


Figure 3.2.4.2-12. Perspective Correction for Pitched View

3.2.4.2.6 Anti-Aliasing and Level of Detail

The aliasing artifacts of ray tracing methods caused by undersampling at long range are completely eliminated by the Step 2 horizontal resampling illustrated in Figure 3.2.4.2-8. In the far zone, the continuous decrease of filter window width with decreasing range smoothly changes the level of detail at which data is displayed. This smooth transition progresses through the mid-zone where the area sampling properties of the Rothstein code method play the anti-aliasing role when filter window size is fixed. In the near zone, continuity equivalent to anti-aliasing is achieved by interpolation. From the discussion on fractals, it is apparent that level of detail at close ranges can be increased without limit, and with no discontinuous changes observable. This property contrasts sharply with polygon models of terrain.

The properties of Steps 1 and 2 discussed above eliminate aliasing effect between horizontally adjacent pixels in the display. Anti-aliasing between vertically adjacent pixels is achieved by the column scanning of Step 3, illustrated in Figure 3.2.4.2-11. The weighted average combining of color at each pixel is equivalent to pixel oversampling in the vertical dimension. The sampling rate is equivalent to the resolution of perspective corrected terrain elevations in pixel units. The combining rule assures correct area times color pixel tone and hence smooth transitions under motion. Step 4 achieves anti-aliasing by the same arguments as given for Step 2, which it closely resembles. Step 5 anti-aliasing is achieved by properties of the Rothstein code discussed in (D.21).

SECTION 4

ADVANCED IMAGE SYNTHESIS SYSTEM DESCRIPTION

4.1 INTRODUCTION

Presented in this section is the functional description of an Advanced Image Synthesis System preliminary design performed as a principal task of this study. This description includes a system summary, performance requirements and detailed characteristics, hardware and software descriptions, and environmental considerations.

4.2 SYSTEM SUMMARY

4.2.1 SYSTEM PURPOSE

The intent of the Advanced Image Synthesis System is to provide a combination of hardware and software resources which will facilitate the development and evaluation of techniques, algorithms, and/or methodologies for generating and displaying visual and sensor scenes in an efficient, cost-effective manner. Subordinate to this objective are desirable system performance characteristics which include but are not limited to the following:

- High quality/fidelity scene generation and display.
- Reasonably high scene generation speed (optimized balance between time and cost).
- Reasonably simple operation, good human interface provisions.
- Large storage capacity.
- Flexibility for readily accommodating new/revised algorithms, techniques, procedures, etc.
- Capability to support efficient data base generation/modification.
- Modular growth/expansion capability (hardware and software).
- Commercial, off-the-shelf hardware.

4.2.2 BACKGROUND CONSIDERATIONS

Image analysis and synthesis functions are usually carried out in one of two types of equipment environments. One is inexpensive, general-purpose,and relatively low in performance. The second is expensive, special-purpose, and high in performance. Standard minicomputer environments fall into the first category; Figure 4.2-1 illustrates the data flow in such a system. Flight simulators based on CIG systems fall into the second category. While large general-purpose computers are sometimes used in large-scale production image processing systems or on a part-time basis in time-sharing environments, these are not cost-effective in flexible experimental environments dedicated to image processing and synthesis. The limitations of each of these categories in terms of cost, performance, and resident software are discussed below.

Standard minicomputer systems usually consist of a medium performance 16-bit CPU, a quarter megabyte of core memory, a disk or two with a few megabytes of storage, a tape drive, and other peripherals. Images are often displayed on calligraphic media such as storage tube CRT's. Refresh memories attached to color raster displays are available at extra cost. While such systems are relatively inexpensive, performance for image processing and synthesis tasks (which constitute the core functions of the system described herein) are woefully inadequate for the following reasons: The 16-bit memory word size is insufficiently accurate for most calculations involved in image generation. The roughly four decimal digits which 16 bits can represent cannot meet both distance and resolution values typical of simulated scenes. For a resolution of 0.1 foot, a distance of less than one mile is the maximum range of data object representation. Matrix operations and perspective division operations are even more critically dependent on sufficient bits of precision.

The core memory size of standard minicomputer systems is barely large enough to hold one black and white toned image of normal television resolution. Since many image simulations require three-color processing, data must often be swapped between disk and memory, slowing down system operation. Existing graphics packages are often very bulky and may need to be partitioned into several core image tasks, further degrading system performance. Such packages are also mismatched to the special characteristics of flight image synthesis. They are often set up for spline-based data bases rather than the grid or polygonal data bases associated with flight simulation. They are designed to simulate special effects such as gloss and translucency which are not important for the purposes of sensor imaging simulation. On the other hand, they are not designed to simulate light sources, haze, fog, and level-of-detail transition — special effects which are very important for sensor simulation. Standard graphics packages designed for general-purpose data structures are inefficient and often inadequate for rendering displays of complex, special-purpose data bases in sensor or visual simulation.

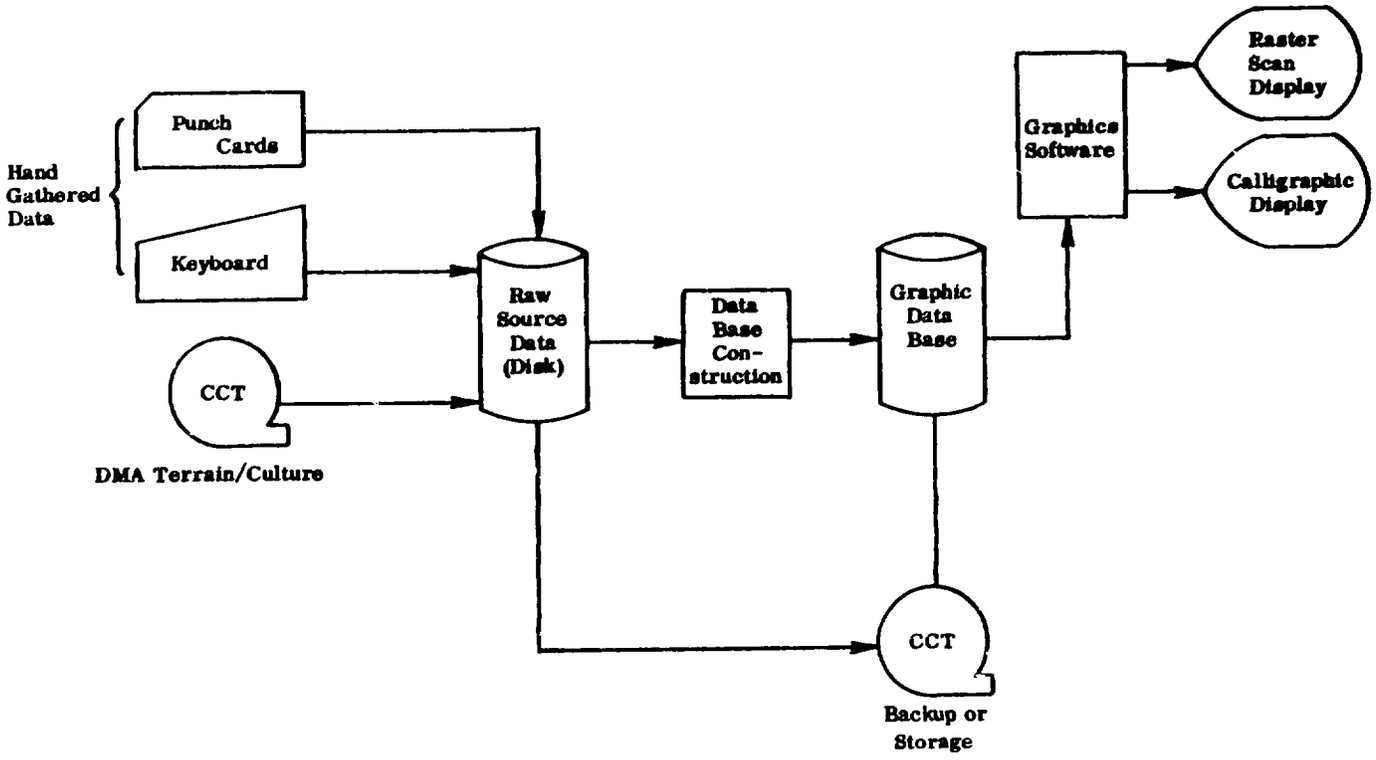


Figure 4.2-1. Data Flow in Minicomputer Based System

The few megabytes of disk storage on standard minicomputers are inadequate for storing large data bases or large numbers of images which the sensor simulation task inputs and outputs, respectively. Ten megabytes will hold only three full-color pictures (8-bit × 3 planes) at 1000 × 1000 image resolution. Magnetic tape is slow to read and write for off-line storage.

Typically, a standard minicomputer-based image generation program may require several hours of run time to produce a single high quality image. However, special-purpose CIG hardware for generating images is expensive (millions of dollars), massive, and inflexible. The hardwiring of algorithms also eliminates the possibility of modifying them to simulate different sensor images and freezes the acceptable data structures to a single type. It is cost-effective to pay this price when it is absolutely necessary to achieve maximum image quality in 1/30 of a second of processing time. Such stringent speed requirements do not apply to the tasks of the sensor simulation system described here.

4.2.3 OVERVIEW OF PROPOSED SYSTEM

The hardware comprising the proposed system will consist of a medium performance 32-bit computer system with about a megabyte of core, a large disk drive, an array processor, and various graphic peripherals as described in paragraph 4.4.1.1. The roughly nine decimal digits of precision can geometrically represent a resolution-to-range ratio which corresponds to 0.01 foot over 2000 miles. This precision is quite sufficient for the image analysis and display tasks which are to be carried out on data whose scale is that of the real world being modelled. The 32-bit floating point hardware in such systems speeds up arithmetic processing. The large memory obviates the need to partition large software packages into swapped tasks and permits direct manipulation of an entire frame of image data. Together these characteristics can yield two orders of magnitude speed-up over standard minicomputers in the performance of image processing and generation tasks. The large number of repetitive indexed array operations provides a favorable environment for an array processor. In summary, one order of magnitude cost increase over existing minicomputers buys two orders of magnitude performance increase. This represents an order of magnitude increase in cost-effectiveness. At the other end of the cost-performance spectrum, CIG systems provide commensurate gains in speed per dollar but at the cost of programming flexibility. The increase in speed is of no particular value for the tasks considered here, and the lack of flexibility greatly limits their usefulness.

The medium-sized disk drive in the proposed system provides on-line storage of large software packages including system utilities such as editors and high level language compilers as well as mathematical libraries, image processing packages, and graphics libraries. The large 300 megabyte disk drive provides space to store a 500 × 500 mile area of DMA Level I data, or 100 full color 1000 line resolution images, or any mix of raw data and images.

High level language applications software will be used to implement sensor image simulation and processing. Sensor image simulation differs from ordinary graphics in that extreme ranges and difficult visibility conditions often prevail. Faithful representation of these conditions plus the unusual characteristics of imaging sensors require special algorithms to be programmed in a manner flexible enough to allow adjustment of parameters to simulate modification of these characteristics. Algorithms for rendering images directly from gridded data bases as well as polygonal data bases must be implemented in the software. Both sets of algorithms must be capable of providing flexibility in rendering the special effects of haze, smoke, meteorological conditions, light, and unique sensor characteristics.

Image processing algorithms can play several roles in the simulation of sensor images. They can be used to simulate contrast enhancement, defocusing, sensor noise, and similar local neighborhood mechanisms. In evaluation on simulated sensor images, photographs of real sensor images can be digitized for comparison with simulated images of the same geographic area. Geometric correction and histogram modification can be used to match the general framework of both images. Edge detection and region segmentation, followed by correlation of the two images can be used to generate quantitative measures of comparison. The digitizing tablet can be used interactively to identify reference points for geometric correction and modify the output of edge and region detection.

4.3 DETAILED CHARACTERISTICS

4.3.1 SPECIFIC PERFORMANCE REQUIREMENTS

The image simulation functions of this system impose unusual performance requirements on several key system components. High computation speeds and mathematical precision are achieved simply by choosing a high performance 32-bit CPU and array processor. Other components of the system are more specialized and are configurable to the particular image simulation tasks which are the primary function of the system. The specific requirements of such components are described below. The key components consist of the imaging system, software, digitizing tablet, and operator/user interface. The last of these combines all the other components in an efficient user conceptual environment.

4.3.1.1 Imaging System

The imaging system shall be capable of providing full color video images as well as high resolution calligraphic images. High speed interfacing between the computer and display devices is required to avoid slow transfer of images between computer and displays.

4.3.1.1.1 The system shall provide RAM refresh memory for each CRT monitor connected to the system.

4.3.1.1.1.1 The refresh memory shall be at least 512 × 512 × 8 bits for each color plane, for each monitor (the choice of a 512 × 512 grid is explained in paragraph 4.4.1).

4.3.1.1.1.2 There shall be at least one color monitor with a shadow mask CRT.

4.3.1.1.1.3 (Desired Option) It should be possible to add three additional planes of refresh memory to the system, each providing 512 × 512 × 8 bits of storage.

4.3.1.1.1.4 (Desired Option) It should be possible to add a second CRT monitor to the system. This monitor can be either monochromatic or color.

4.3.1.1.2 The system shall provide the capability of outputting and inputting images of the line drawing ("wire frame") type.

4.3.1.1.2.1 There shall be at least one monochromatic calligraphic monitor.

4.3.1.1.2.2 The calligraphic monitor shall have an attached light pen for inputting data.

4.3.1.1.3 (Desired Option) It should be possible to quickly produce a hard copy of any image viewed on the color monitor.

4.3.1.1.3.1 (Desired Option) The system shall include a Polaroid film recording device.

4.3.1.1.4 (Desired Option) It should be possible to record and play back a sequence of synthesized images (e.g., snapshots along a route).

4.3.1.1.4.1 (Desired Option) The system shall include a magnetic video disk recorder and playback unit. The unit should be capable of recording and playing back analog signal images of 512 × 512 × 8 bits × 3 planes.

4.3.1.1.5 (Desired Option) It should be possible to digitize a color photograph and input it to the system.

4.3.1.1.5.1 (Desired Option) The system shall include a color image scanner of at least 8 bits per pixel per color resolution.

4.3.1.1.6 The system shall be capable of receiving commands sent from the host computer via the interfaces described in paragraph 4.3.1.1.7 and processing the commands for each monitor connected to the system.

4.3.1.1.6.1 These commands shall allow a program resident in the host computer to execute all functions performed by the monitors.

4.3.1.1.6.2 Commands shall exist for allowing the host computer to read and write the refresh memory directly.

4.3.1.1.7 All monitors shall be capable of being interfaced to the host computer.

4.3.1.1.7.1 The connection of the display system to the host computer shall use the interface described in paragraph 4.4.1.

4.3.1.1.7.2 The interface shall permit a program running on the host computer to read from and write to the refresh memory using the host computer's direct memory access facilities.

4.3.1.1.7.3 The interface shall permit a program running on the host computer to control or initiate commands which cause the display system to perform its functions.

4.3.1.2 Software

The bulk of the software shall be written in a high level language such as Fortran in order to facilitate readability, flexibility and portability.

The software shall conform to the standards of good design principles in order to optimize troubleshooting, maintenance and future extension of capabilities. These standards shall include a structuring into modules, each of which has a unique function and a clearly specified interface to modules which reference it. Duplication of functions shall thereby be avoided and isolation of errors facilitated. The organizational, functional, and performance requirements of the software are outlined below.

4.3.1.2.1 The software shall be organized into a hierarchical structure under the control of an executive module which selects lower level modules for execution based on operator commands. The executive shall be structured in such a way that new commands can be easily added at any time.

4.3.1.2.1.1 Each generator command shall invoke a distinct command servicing module.

4.3.1.2.1.2 Each command servicing module shall call a sequence of applications and utility modules.

4.3.1.2.1.3 The function of applications modules shall include three general categories: Computer Graphics, Image Processing, and Mathematical Libraries. These modules shall perform computational and geometric functions on mathematically defined data.

4.3.1.2.1.4 Utility modules shall consist of device drivers, error handling routines, operator interfaces, and systems software such as editors and high level language compilers. Some of the device driver modules may be written in low level languages for performance efficiency.

4.3.1.2.2 Software utility modules shall be provided to perform the following operations on the refresh memory associated with each independent monitor.

4.3.1.2.2.1 It should be possible to erase any plane of the refresh memory.

4.3.1.2.2.2 It should be possible to randomly read to or write from any element of refresh memory.

4.3.1.2.2.3 It should be possible to mix alphanumeric text and graphics on a monitor.

4.3.1.2.3 Software utility modules shall be provided to perform the following operations with the calligraphic display.

4.3.1.2.3.1 It should be possible to draw a randomly sized and positioned vector on the calligraphic display in less than 35 microseconds.

4.3.1.2.3.2 A drawn vector shall be specified by the coordinates of its endpoints at arbitrary positions on the screen.

4.3.1.2.3.3 The system shall accurately represent vectors whose endpoints are specified outside the physical coordinates of the calligraphic display without wraparound.

4.3.1.2.4 It should be possible to use the proposed system to create and display a sensor or visual scene.

4.3.1.2.4.1 It should be possible to display a raster scan scene of at least 5000 edges in under 30 seconds.

4.3.1.2.4.2 It should be possible to create a scene constructed from polygons to an accuracy of 1/32 (one thirty-second) of a foot.

4.3.1.2.4.3 It should be possible to create scenes of at least $(64)^3$ colors and 32 distinct textures.

4.3.1.3 Digitizing Tablet

The digitizing tablet shall provide a means of capturing geometric data from maps, drawings or other hard copy unit. It shall also provide a means of moving a cursor

on a display screen, tracking the user's motion of a stylus on the tablet in real time. The tablet shall capture position data with high precision and repeatability. No awkward or difficult hand coordination shall be required of the user in operating the tablet.

4.3.1.3.1 It shall be possible to interface the host computer to a digitizing tablet.

4.3.1.3.1.1 It should be possible to connect the host computer to the digitizing tablet and sense and control the status of the digitizing tablet.

4.3.1.3.1.2 It should be possible to determine the tablet's stylus position and status.

4.3.1.3.2 The tablet work surface shall be operable at any angle from zero to ninety degrees, shall function with source document(s) placed on the surface with the pen and cursor above the document, and shall be designed for ease and convenience of use.

4.3.1.3.3 The tablet working area may be either square or rectangular, but in either case must be of sufficient size to accommodate the largest source document anticipated to be digitized. The entire surface area shall be usable under software control.

4.3.1.3.4 The resolution of the tablet shall be at least 0.01 inch. Accuracy, linearity, repeatability and stability shall be ±0.005 inch or better. The tablet shall function (with diminished performance) with up to 0.05 inch of nonmetallic material placed upon it.

4.3.1.3.5 The tablet shall be equipped with a pen which shall be activated by pressure on the tip. The amount of pressure required shall be such as not to produce fatigue during long periods of use, yet sufficient to avoid any ambiguity in its status. Tactile feedback indicating activation is desired. The pen shall be usable at any angle to the table between 30 degrees and 90 degrees.

4.3.1.3.6 The tablet shall be equipped with a movable cursor which shall be constructed in such a manner that parallax is minimized. The cursor shall have integral buttons for programming. These buttons shall produce appropriate signals available through the interface to the computer.

4.3.1.3.7 Data entry selectable via pen or cursor activation shall include single coordinate pairs (points) and continuous coordinate pairs (tracks). The entry of continuous coordinate pairs (moves) while the pen or cursor is in the proximity of the surface shall also be accommodated.

4.3.1.4 User Interface

The software and hardware shall be integrated together to provide the user with a habitable easy to use environment conforming to good human factors standards, outlined below.

4.3.1.4.1 The user shall control the system by issuing commands which correspond naturally to the tasks to be performed. Computer operating system commands or programming are to be strictly avoided except for initiating or modifying the system.

4.3.1.4.1.1 Commands may be entered from the keyboard or by selecting a command from a menu via the digitizing tablet or light pen.

4.3.1.4.1.2 The system shall provide prompts, queries, and displays of current processing status so that the user is never ignorant of his options.

4.3.1.4.2 All geometric tasks involving selection of a position, drawing a feature, placing symbols on displays and the like, shall be carried out by exploiting the eye-hand coordination provided by the digitizing tablet stylus tracked by a display cursor.

4.3.1.4.3 The user shall always have control over major processing events without resorting to the computer operating system.

4.3.1.4.3.1 The user shall be able to abort a long running program without crashing the system.

4.3.1.4.3.2 The user shall be able to evoke display of HELP files for guidance.

4.3.1.4.3.3 A logging can be invoked which records on files, the history of user actions and values of parameters (such as viewpoint) during a work session. Any subset of the sequence can be used to replay the processing just as if the user reenacted his commands.

4.3.1.4.4 The format of operator inputs and system prompts shall be identical for similar actions under different command contexts.

4.3.2 SYSTEM FUNCTIONS

The proposed Advanced Image Synthesis System will be used to perform two primary functions: (1) data base development, and (2) data base display. Each of these functions is described separately in the following paragraphs. A diagram of the data flow for both functions is provided in Figure 4.3.2-1.

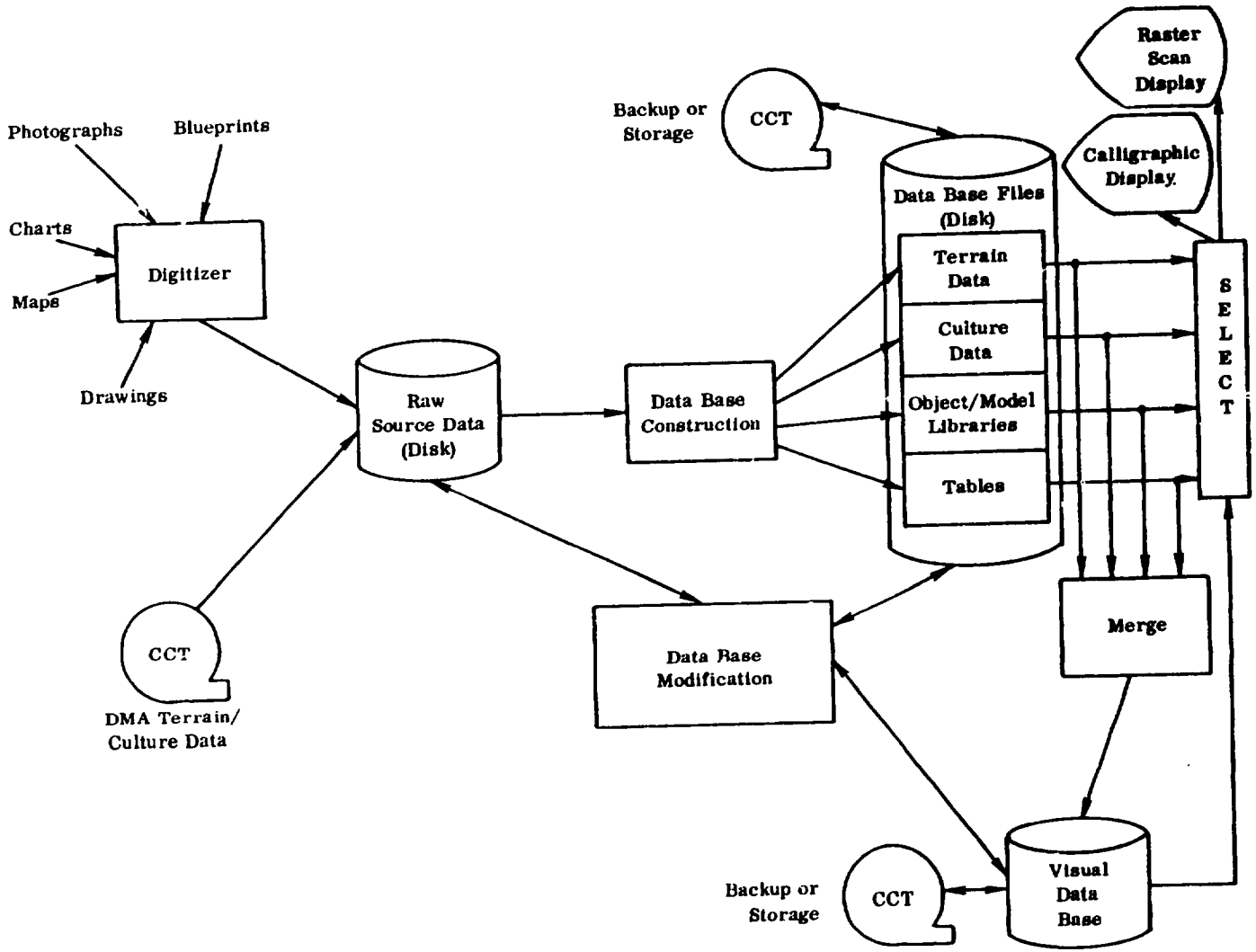


Figure 4.3.2-1. Data Base Development System Flow Diagram

4.3.2.1 Digital Data Base Development

The data base development operation shall consist of four major tasks, as follows:

a. Intermediate Data Base Construction:

This operation shall be performed by routines which accept data from different sources and generate intermediate data bases from them. In order to maintain system efficiency and flexibility, the operation of each of the processing routines should be well-defined and separate from each other. The files produced by these routines are intermediate files which will be formatted and combined in whole or in part into a final visual data base. Some of the subtasks involved in this operation are described below.

(1) Terrain Data Base Development - A terrain data base will consist of a network of terrain triangles. The source data will consist of DMA terrain files, contour maps, and/or surface-specific points. Routines can be developed for automatically processing DMA data to form a triangular surface. Contour maps can be input to the computer by mounting them on a digitizing table and having the operator trace the contour lines with a digitizing stylus. After each contour line is entered, the operator will type in the line's elevation. Software will provide the capability of constructuring a triangulation from a contour map.

The system user can also input terrain data from maps, drawings, and aerial photographs. The user will mount and register such media onto the digitizing tablet; he will then point to specific terrain points with the stylus and enter their respective elevation via keyboard. Ridge segments will be input by their endpoint pairs. A triangulation will then be automatically formed over these points and segments.

(2) Culture Data Base Development - Culture data may be automatically input to the computer from DMA culture tapes. Alternatively, cultural information can be digitized from maps, charts, drawings, or photographs. The source material will be mounted on the digitizing tablet, and the polygonal culture areas traced with a stylus. The operator will type in the required descriptive parameters.

(3) Object/Model Data Base Development - The data base development system will have the capability to generate and maintain Object and Model Libraries. The Object Library will consist of a directory accessible accumulation of data sets, where each data set completely defines a three-dimensional object. The Model Library, similarly

to the Object Library, will consist of a directory accessible accumulation of data sets, where each data set is referred to as a model. An object shall represent a single generic feature such as a tree, building, or geometric solid. A model shall consist of a collection of objects arranged in some arbitrary configuration. This building-block hierarchy of objects and models assures efficiency of storage and manipulation. A model need be defined only once; it may, however, be placed at a number of locations within the mission environment. Each copy can be given a separate orientation and scaling.

(4) Table Entry - Several tabulated sets of data will be included in the Visual Data Base file. These tables will contain such information as the parameters governing the color and texture of scene components. For example, a culture polygon may have a code indicating that it represents a deciduous forest. This code will be a pointer to table entries defining the color and texture of deciduous forest. Part of the visual data base creation process will be the definition, generation, and modification of tabulated data. The table generation process will include both batch and interactive capabilities.

Several alternative tables may be available for the same scene. They will allow the operator to display the scene as it would be seen on different sensor displays or under different visibility conditions.

To further define the intermediate data base construction function in terms of specific user commands, system queries, user inputs/responses, and user actions, the following details are provided:

Command: INITIALIZE TABLET

Description: This command initializes the digitizing tablet for data entry. The associated software is placed into a data entry mode. The calligraphic screen is erased and readied to display the points and line segment input through the tablet.

Command: TABLET SETUP

Description: This command sets up the digitizing tablet for data entry. The user mounts a map, chart, or overhead aerial photograph onto the digitizing tablet, then the mounted sheet is registered by touching the stylus to four points on it (usually the corners) and typing-in respective latitude/longitude or x/y values. The user specifies the map transformation to be used (Mercator or linear). The data is stored in x/y coordinates with a user-specified origin.

System Query	User Input or Response	Typical Action by User
Touch stylus to first registration point.	User places the stylus over the first registration point.	User touches the bottom left corner of a map.
Location.	User types in the latitude and longitude of the first registration point.	User types 30°20'/44°11'.
Transformation type.	User types in the type of transformation to be used.	User types in linear.
Touch stylus to origin; i.e., x = 0, y = 0 point.	User places the stylus over the desired coordinate origin.	User touches the stylus to the map's approximate center.

Command: MODE

Description: This command prepares the system to accommodate data of a certain mode. The allowable modes are: vertex, edge, face, object, polygon, contour line, and object location.

System Query	User Input or Response	Typical Action by User
Data mode.	User types in the desired data mode.	User types in vertex.

Command: ENTER

Description: This command allows the user to enter data through the digitizing tablet, and is used subsequent to the following sequence of operations: First, the tablet is INITIALIZED and SET UP, and then a map, chart, drawing, or blueprint is placed onto it and registered. Following this, a data mode is selected by running the MODE command.

Next, the user places his stylus at some desired location on the surface of the digitizing tablet. When he presses the button on the stylus, that data location is sent to the computer. The user indicates that he is through entering data by pressing the button on the stylus three times in quick succession.

All data entered via tablet is displayed at the calligraphic CRT. If the system is in the vertex mode, then entries are displayed as x's. Otherwise, the entries are displayed in the form of the line segments joining the input locations. The calligraphic display is scaled to correspond to the working area of the digitizing tablet.

System Query	User Input or Response	Typical Action by User
[if in Terrain Mode]		
ENTER 1ST VERTEX and ELEVATION ⋮	User touches the stylus to the first terrain surface specific point and types in its elevation.	User touches the stylus to the peak of a mountain, and types in 1000 at keyboard.
ENTER ENDPOINTS OF 1st RIDGE SEGMENT and ELEVATIONS ⋮	User touches the stylus to the two endpoints of a ridge line segment, then types in two elevations at the keyboard.	User touches the stylus to the two endpoints of a cliff appearing on a map mounted on the tablet, then types in 1000, 1010 at the keyboard.
[if in Object Location Mode]		
ENTER 1st LOCATION and NAME of OBJECT ⋮	User touches his stylus to some location on a mounted map. Then he gives the name of the object to be placed there, by entering the name at the keyboard.	User sees a mark on a mounted map indicating the location of a water tower. He places his stylus at that map location and types in the words WATER TOWER at the keyboard.
[if in Contour Line Entry Mode]		
ENTER 1st CONTOUR LINE and its ELEVATION ⋮	User traces a contour line with his stylus and then types in its elevation at the keyboard.	User has mounted a contour map on the tablet. He traces a contour line with his stylus and types in 1020.
[if in Culture Mode]		
ENTER VERTICES	User touches the stylus to the vertices of a culture polygon.	A map is mounted on the tablet, and the user touches the stylus to 12 points on border of a lake.
ENTER ATTRIBUTE	User types in some attribute indicating material type.	User types in WATER.

Command: TRIANGULATE

Description: The operator first types in the name of the temporary file containing the data to be triangulated. Next, he types in the name of the file to hold the computed triangular network. The system then activates the appropriate software routines to process the given data type. The input data file can be one of three types: (1) points and line segments, (2) digitized contour map, and (3) raw DMA terrain file.

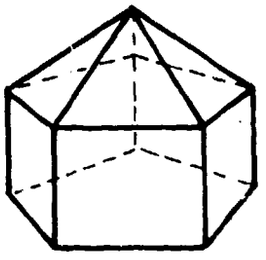
System Query	User Input or Response	Typical Action by User
Source File	User types in the name of the temporary file holding the data to be triangulated.	User types in TRIDAT.
Destination File	User types in the name of the file to hold the computed triangular network.	User types in TRINET.

Command: BUILD

Description: This command provides a convenient method for generating three-dimensional objects. The desired shape is specified in simple terms and then constructed in software.

The user constructs objects the way in which a building is put together. After the shape is defined at each level, the system then builds the base, walls, and roof.

For example, to define a tent with a pentagonal base, three levels must be defined. The base is five sided, with the roof converging from five sides to a point.



The eleven object vertices are computed in software along with the faces for the base, walls, and roof.

Query	User Input	Typical Action by User
Give object name.	User specifies a name for the object of his creation.	User types "tent".
Name destination file.	User types in the name of the file where the object is to be put, most likely an object or model library.	User types OBJLIB.
Level 1: No. of vertices, radius, height.	User types in the number of vertices, radius, and height for the first level of the (symmetric) object.	User types: 5,50,0
Level 2: Number of vertices, radius, height. . : :	User types in the number of vertices, radius, and height for the second level of the object. (A carriage return is typed for an exit.)	User types in: 5,50,20

Command: CHECK

Description: This command checks a locked-on feature to see if it is geometrically and topologically consistent. Convexity is also tested for when appropriate.

Typical conditions tested for are as follows: A vertex must have two coordinates if it is part of a planar feature and three if it is part of a three-dimensional object. A polygon must have at least three vertices. The edges of a polygon should not cross (i.e., the polygon must be simple). A polyhedron must enclose volume.

The result of this command is a report printed at the user's console.

Command: LOAD

Description: This command loads data stored in a temporary file into an intermediate file. For example, a culture polygon entered through the digitizing tablet may be stored in a file named TEMP. The user can take the data from TEMP and place it in an intermediate culture file names CULT1. The x/y coordinates of the data remains unchanged.

System Query	User Input or Response	Typical Action by User
Name of source file.	User types the name of the temporary file to be loaded.	User types in TEMP.
Name of destination file.	User types the name of the intermediate file.	User types in CULT1.

b. Data Base Modification:

Several data files intermediate between raw input data and data suitable for final sensor simulation display will be created during data base development. In order to correct deficiencies noted during display or to add to the intermediate data bases, each intermediate terrain or culture type will be capable of being modified. Data base modification will also be necessary for missions where incoming intelligence information specifies new locations of ground objects (e.g., tanks). The user will have the ability to modify the makeup and placement of objects interactively when practical; that is, he will be able to point to features on the display using a digitizing stylus driven cursor. Features so identified will then be modified according to the commands the user inputs.

As was done in the intermediate data base construction operation previously described (paragraph 4.3.2.1a), the following specific operator commands, system queries, operator inputs/responses, and operator actions associated with the data base merging functions are provided:

Command: LOCK (UNLOCK)

Description: The LOCK command allows the user to lock onto a feature displayed on the calligraphic CRT. The current system mode determines the type of feature locked onto. After the user places his light pen onto the calligraphic display, the system reads the pen position and locates the nearest component feature of the current type being considered. The component flashes on and off to indicate that it is locked onto. For example, if the system is operating in a vertex mode, the nearest vertex to the light pen position will be locked onto. Only one component can be locked onto at a time. (The UNLOCK command frees the locked feature.)

System Query	User Input or Response	Typical Action by User
Point light pen to feature.	User points his light pen to a feature to be locked onto.	User points his light pen to a vertex of a culture polygon.

Command: LOCATE

Description: This command allows the user to find the coordinates of any locked-on feature. If the feature is a single vertex, its two or three coordinates will be printed at the user's console. If the feature is an edge, polygon, or polygonal face, the defining set of coordinates will be printed. If the feature is an object, the set of coordinates defining each face will be listed separately.

Command: MOVE TO

Description: This command allows the user to move a locked-on feature to a new x/y location of his specification simply by typing in the new location. The centroid of the new feature is then translated to the new location and the calligraphic display and data file are updated accordingly.

If no feature is locked-on, the user can specify the location at which a library object or model is to be placed. He identifies the particular object to be retrieved by name.

System Query	User Input or Response	Typical Action by Use
Type new location.	User types in a new location for the centroid of the locked on feature.	User types in: 40, 80, 90
(If no feature is locked on)		
Type name.	User types the name of the library object or model to be placed at the indicated location.	User types "HOUSE".

Command: DELETE

Description: This command allows the user to delete any locked-on feature. If the feature is a vertex, the vertex is deleted and the succeeding and preceding vertices are joined. The same thing happens for an ed-

If the locked-on feature is the face of a three-dimensional object, the face is deleted. If an entire polygon or 3-D object is locked-on, the entire feature is deleted. In each case, the calligraphic display and data file are updated to indicate the change. If the entire contents of a data file have been deleted the file itself is wiped out.

The user is given a chance to renege. After a feature is deleted, the user is asked to confirm his action. If he types "no", the display and data file are returned to their previous form.

System Query	User Input or Response	Typical Action by User
Do you confirm deletion?	User types "yes" or "no".	User types "yes".

Command: ATTRIBUTE

Description: Each polygon or polygonal face will have an attribute associated with it. The attribute is in the form of a name, usually indicating material composition.

This command causes the system to type the name of the locked-on feature's attribute.

For example, a polygon may have an attribute of "swamp". A polyhedral face may have an attribute of "brick". A contour line may have an attribute of "300 feet".

The user is given the opportunity to change the feature's attribute name. A null response indicates no change. An affirmative response is indicated by the user typing in a new attribute name.

System Query	User Input or Response	Typical Action by User
Type new attribute name.	User hits carriage return to indicate that the attribute name is not to be replaced. Otherwise, he types in the name of the new attribute.	User types in "grass".

Command: RESCALE

Description: This command allows the user to change the size of a locked-on feature. The operator just types in two or three elements of the scaling vector. The locked-on feature is then rescaled, with its centroid left in place.

The calligraphic display and data file are updated to indicate this change.

System Query	User Input or Response	Typical Action by User
Type scale parameters.	User types in two parameters if the locked on feature is a polygon and three if it is a three-dimensional object.	User types in 1.1, 0.75

Command: SEARCH

Description: This routine allows the user to search through the contents of an Object or Model Library. The objects when reached are displayed calligraphically, one at a time. The routine is exited when the user types a carriage return. Upon exit, the last displayed object remains displayed on the calligraphic CRT.

System Query	User Input or Response	Typical Action by User
Source File.	User types in the name of the object or model file to be searched.	User types in OBJ1.
Continue Search?	User types a "Y" or "YES" to place the next object in the file on the screen. A typed "N", "NO", or carriage return causes the command routine to be exited.	User types a "Y".

Command: TRANSLATE

Description: The TRANSLATE command fits into the following sequence of operations: The user first plots the data stored in some file on the calligraphic display. Then, he locks onto some component, e.g., a vertex, line segment, face, or entire object. The user next types TRANSLATE. He indicates the translation by either typing in a $(\Delta x, \Delta y)$ value at the keyboard or by placing the light pen on a new location on the calligraphic display. The system will be equally ready to accept either type input once this command is entered. The data in its new form is stored and displayed on the calligraphic CRT.

System Query	User Input or Response	Typical Action by User
Indicate translation.	User either types in a translation vector or points with the light pen to a new location on the calligraphic CRT.	User type in: 45, 60.

Command: ROTATE

Description: This command allows the user to rotate any locked-on polygon or object about the x, y, and/or z axes. The rotation takes place about the object's centroid, using a right hand coordinate system.

The calligraphic display and data file are updated to indicate this change.

System Query	User Input or Response	Typical Action by User
Type rotation angles θ_1, θ_2, θ_3 about x, y, and z axes in degrees.	User types in three angles.	User types in: 0, 0, 45.

Command: STATISTICS

Description: This command causes a listing of the data within any named temporary, intermediate, or final data base file. The printout will be in the following form.

CULTURE:	Number of culture polygons: Number of culture edges: Number of culture vertices: Number of culture attributes:
TERRAIN:	Number of terrain triangles: Number of contour lines: Number of terrain edges: Number of terrain vertices:
OBJECTS:	Number of objects: Number of faces: Number of edges: Number of vertices:
TABLE:	Number of table entries:

Query	User Input	Typical Action by User
File name.	The user types in the name of the file for which he wants the statistics printed.	The user types in CULT1.

c. Data Base Merging:

The visual data base will be produced by merging the following intermediate data types: (1) triangular terrain network, (2) polygonal culture data, (3) object/model library, and (4) tabulated data. The system user will type in the name of the files to be merged, then the merging process will be automatic. The merging routines will inform the user of any conflicts in data definition; for example, an attempt to place two objects onto the same location.

d. Display of Intermediate Data:

Since the data base development process is rather complex and operates on inherently geometric data, the user must have access to visual displays of intermediate data. These displays provide efficient assessment of intermediate processing and a means for rapidly locking on to features to be modified via digitizing tablet control of a displayed cursor. The display of intermediate data need not completely replicate the final sensor simulation display. In fact, a calligraphic (wire-frame) display may provide all the geometric reference cues necessary to carry out many of the data modification tasks in a fraction of the processing time required to generate a raster image.

To conform to good human factors design, the actual display commands will be consistent across all intermediate data types being displayed. The entered command routine will cue the user to supply the type of parameters required for the type of data being displayed. This requirement implies the existence of data formatting and conversion routines to serve as an interface between the display commands and the data.

Specific user commands, system queries, user inputs/responses, and user actions associated with the "display of intermediate data" function are as follows:

Command: DRAW

Description: The calligraphic screen will be erased once this command is entered. This command causes any named data file to be drawn at the calligraphic CRT.

The user is asked to enter a viewpoint, boresight, and field of view. If he does, a true "wire frame" drawing results.

A null response to this query indicates that an overhead view is desired. The overhead view is scaled to fit the display. One exception occurs when the locked-on feature is a single three-dimensional object. For this case three orthographic projections are plotted.

Any isolated vertices in a data file are plotted as small x's.

System Query	User Input or Response	Typical Action by User
Name data file.	User inputs the name of the data file to be plotted.	User types CULT1.
Input: viewpoint, boresight, and field of view in degrees.	User either types in the required information or hits carriage return.	User types: -100, 50, 250; 0, 0, 0; 45

Command: SCAN

Description: This command causes any named data file to be displayed on a raster scan CRT as a matrix image. The user is asked to enter a viewpoint, boresight, and field of view. These parameters are required to define the perspective display.

The user has the option of choosing fast or slow display generation. A fast display does not implement edge smoothing. A slow display produces a higher quality image, but takes about 16 times longer to run.

System Query	User Input or Response	Typical Action by User
Name data file.	User inputs the name of the data file to be scanned out.	User types DROP _ ZONE.
Input viewpoint, boresight, and field of view in degrees.	User types in the requested viewing parameters.	User types: -120, 30, 230; 0, 20, 0; 30.
Fast or slow display?	User types an "F", the word "FAST", or a carriage return to indicate a fast display; or an "S" or the word "SLOW" to indicate a slow display.	User hits carriage return.

4.3.2.1.1 Interactive Data Base Development and Display Commands

Each of the four data base development and display tasks will be controlled by user commands. These commands may be evoked via keyboard or tablet menu. The system will provide prompts and queries to guide the user through complex sequences of actions. An abort mechanism will be provided to permit the user to terminate any process, even if it is in progress. If the user attempts something which is not allowed (e.g., enter out-of-range data), an error message will be printed, and the command exited.

A block diagram showing the organization of the system commands and utility routines is given in Figure 4.3.2.1.1-1.

4.3.2.1.2 Utility Routines

A number of functions are required to support the Visual Data Base development tasks. These include:

(1) Operator Interface
(2) Device Drivers

- Calligraphic CRT
- Raster Scan CRT
- Digitizing Tablet

(3) Data Handlers

- Data Format and Manipulation Routines
- File Managers

(4) Command Files

All the above modules will be subordinate to the System Executive. This will allow a top-down structured software design, as well as enable the development of a controllable flexible, expandable system. The System Executive will enable the user to communicate with the general purpose computer's operating system and the data base development software only through the interactive command structure. Calls to utility routines will be transparent to the user.

4.3.2.2 Data Base Display

The user will be able to specify any viewpoint, attitude and field of view for both perspective and orthographic projected image displays of geometric data. Default values will be assigned for viewing parameters which are not specified by the user. There will be two distinct display formats for viewing geometric data such as models,

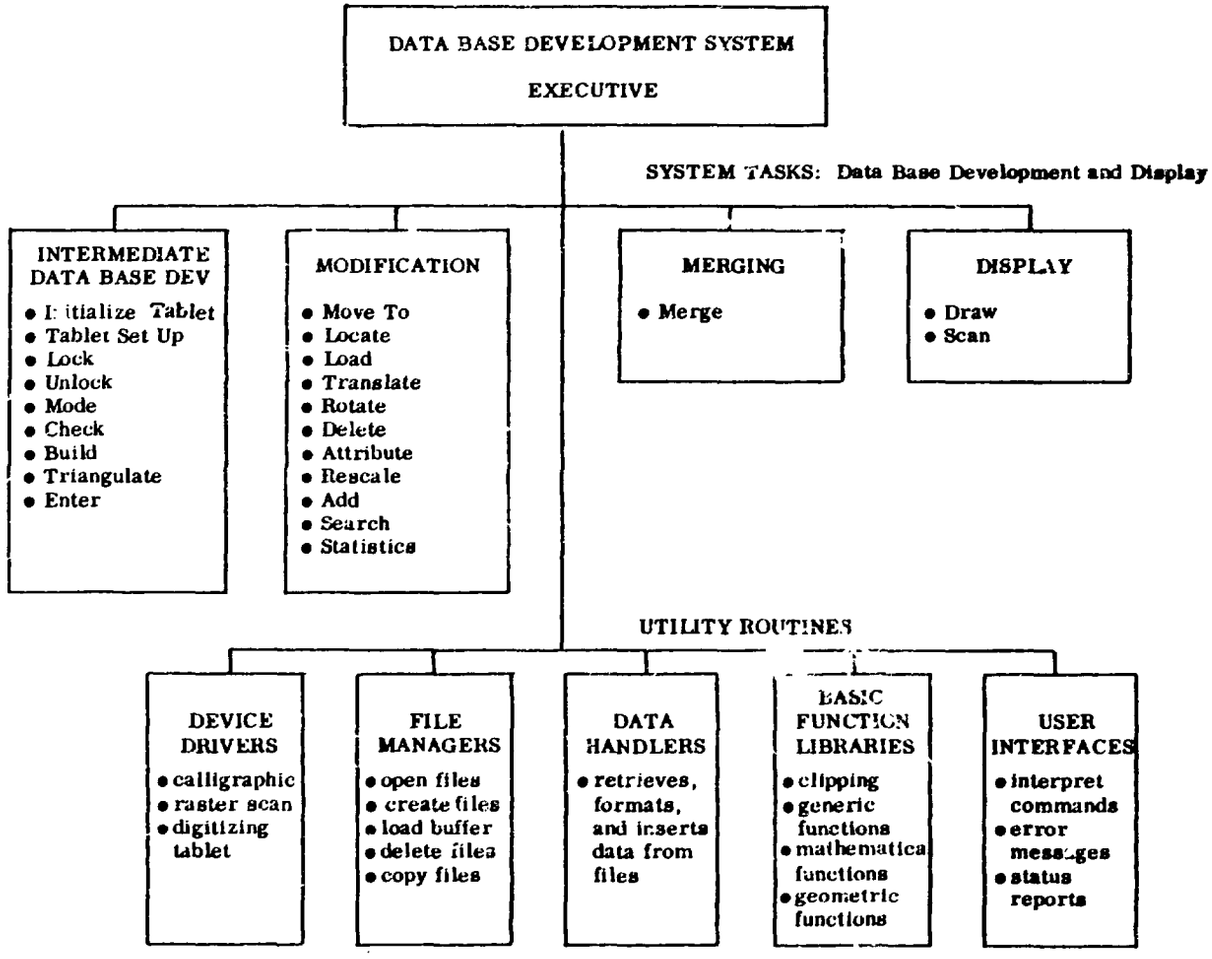


Figure 4.3.2.1.1-1. Block Diagram of Software Organization

intermediate files and scenes. The first is a calligraphic format in which data is depicted as a "wire-frame" drawing consisting of a set of straight lines of uniform color and thickness displayed against a blank background on a CRT. The software module which will generate such a display will be called DRAW. Its relationship to other system functions is shown in Figure 4.3.2.1.1-1. DRAW will call a standard calligraphic plot utility routine to join the pairs of vertices forming the edges of a scene.

The second type of display format is based on raster scan image rendering under control of the SCAN command software. This type of image is defined by specifying the color of each of 512×512 pixels of a color video screen. The resulting image can depict continuously shaded, colored surfaces. The image can faithfully represent surface reflective properties ranging from opaque to transparent and shiny (metallic) to dull (matte). Generating color raster images requires considerably more computation time and memory than generating calligraphic images. However, raster images are essential to correctly simulate most sensor images or visual scenes. The basic processing modules comprising the raster scan display function and the data flows between them are shown in Figure 4.3.2.2-1. Each module is briefly described below.

Perspective Projector - The Perspective Projector will select those edges from the original 3D data base which appear within the field-of-view (a clipping operation). The perspective projection of each edge in the data base shall be computed. This module shall also adjust face colors by calculating illumination as a function of sun/moon position relative to face orientation.

Priority Processor - The Priority Processor shall use viewpoint, separation plane data, centroid locations, and relative priority numbers to compute a unique absolute priority number for each face in the environment. The faces shall be listed in priority order, with back-facing faces excluded.

Edge Generator - The Edge Generator, which shall receive edge data from the Perspective Projector, shall store potentially visible edges. It shall calculate the raster line intercept values from the two endpoints of each potentially visible edge. For each scan line, the Edge Generator shall determine which edges intersect the top and bottom of the line.

Edge Orderer - The Edge Orderer shall receive intercept values and edge words from the Edge Generator. The Edge Orderer shall sort the potentially visible edges in left-right order for each scan line.

Priority Resolver - The Priority Resolver shall receive potentially visible face data from the Priority Processor. The Priority Resolver shall create a face priority list in which face priorities are addressed by face number. For each raster line, the Priority Resolver shall receive potentially visible edge data from the Edge Orderer. For each edge, the edge's face number shall be used to obtain from the face priority list the relative priority number of the edge.

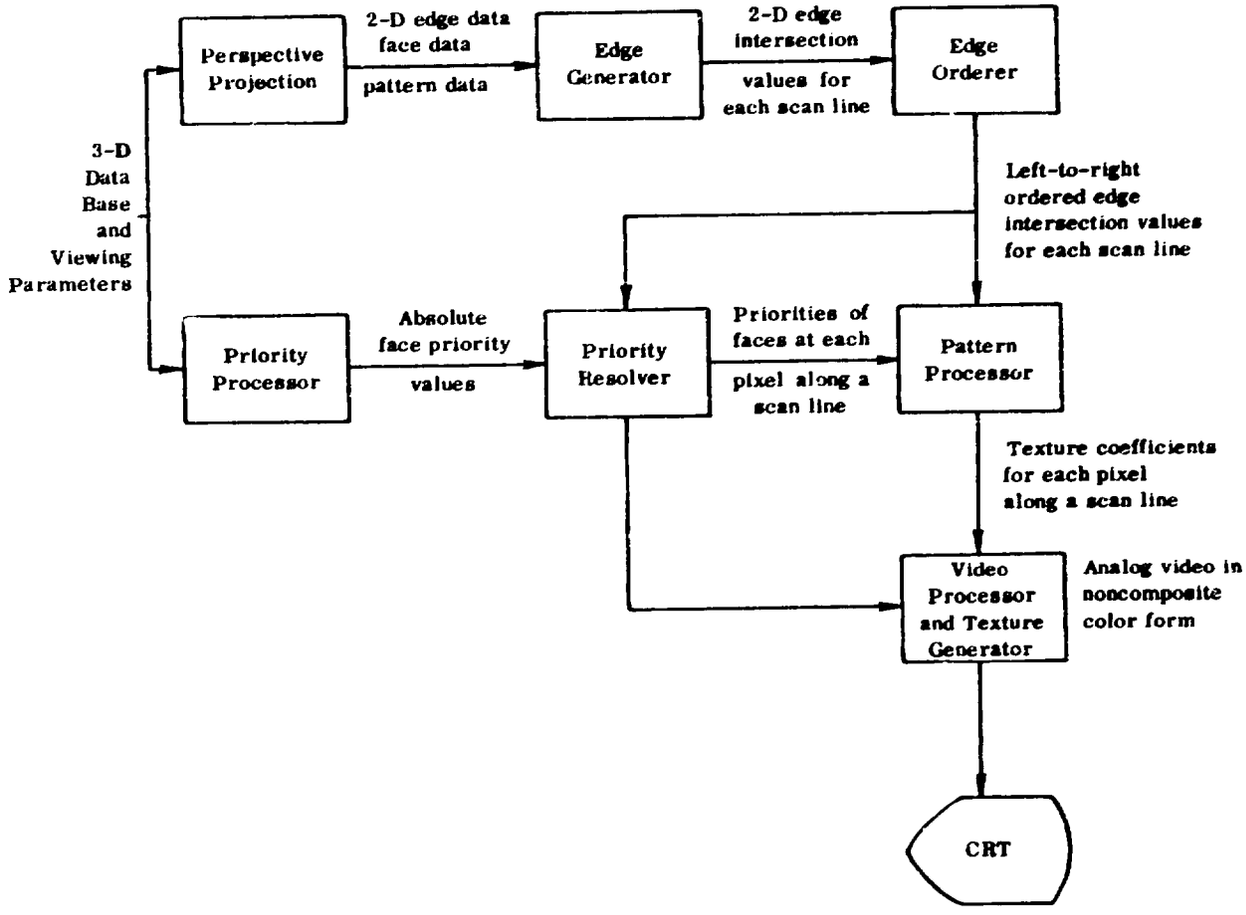


Figure 4.3.2.2-1. Raster Scan Display Functional Flow

Pattern Processor - For each scan line, the Pattern Processor shall receive left and right position limit values for each scanned face along with the face number. Texture data shall be obtained for each face and texture coefficients shall be computed.

Video Processor and Texture Generator - The Video Processor shall receive visible edge words from the Priority Resolver and texture data from the Pattern Processor. A set of red, green, and blue intensity values shall be computed for each pixel based upon the weighted colors of the faces which project onto the pixel. This color shall be modified by texture values and a surface fading function.

4.3.3 INPUTS AND OUTPUTS

The inputs to the proposed system shall be DMA terrain and culture tapes, maps, charts, drawings, blueprints, photographs, and intelligence information. The output of the system will be digital and calligraphic imagery. An image can be viewed as a picture on a CRT monitor, photographed, or placed onto magnetic tape.

4.3.4 DATA CHARACTERISTICS AND STORAGE

The heaviest demands on core storage will be imposed by the image synthesis software. Two fundamentally different approaches to image generation will be employed. In one, the structure of the data base represents a regular grid of elevation samples; in the other it represents a network of polygons.

When using a grid data base to generate an image, it is often desirable to keep the generated scene entirely in core memory. The storage requirement for a $512 \times 512 \times 2$ byte image is about 500 kilobytes. Other storage requirements for data, programs, and working space increase the needed storage to around 750 to 1000 kilobytes.

A good polygon data base scene generator should be able to process a maximum of 10,000 edges. The storage requirement for edges and associated parameters is about 300 kilobytes. The executable code plus storage space for local arrays adds another 400 kilobytes to the memory burden. Another 100 kilobytes should be available for system overhead. Thus, the total memory requirement for an edge-oriented scene generator is about 800 kilobytes. In total, the computer core storage requirement is approximately 800 to 1000 kilobytes.

Archival storage will be achieved with magnetic tapes and disks. A 67 megabyte disk is standard for system use; however, a 300 megabyte disk is suggested for image and data storage.

4.4 ENVIRONMENT

4.4.1 EQUIPMENT ENVIRONMENT

4.4.1.1 System Description

The proposed Advanced Image Synthesis System is composed of off-the-shelf equipment configured as shown in Figure 4.4.1-1. The two major subsystems are the computer and the graphic work station; these are comprised as follows:

Computer Subsystem	Graphic Workstation
1 - medium-size, 32-bit computer system (with 512 kilobytes to 1 megabyte of core storage)	1 - digitizing tablet and stylus
1 - array processor (optional)	1 - color CRT monitor (with 3-to-6 channels of refresh memory)
1 - line printer	1 - black and white CRT monitor (with refresh memory)
2 - disk drives (one 67 mb and one 300 mb)	1 - calligraphic display and light pen
2 - nine-track magnetic tape units	1 - Polaroid film image recorder (optional)
1 - system console	1 - magnetic video disk (optional)
Up to three interactive terminals	1 - color image scanner/digitizer (optional)

The Digital Equipment Corporation VAX computer or equivalent medium-scale mainframe is recommended. Core and disk storage capacities are based on the performance criteria discussed in paragraph 4.2.3. The graphic equipment may have to be purchased specifically for this application. The graphic work station (Figure 4.4.1-2) can initially be configured as a core system, with optional equipment added at a later date. This core system will be sufficient for data base development, display, and processing and all other system tasks. The optional equipment will add the capabilities of inputting and outputting hard copy color photographs and animated sequences.

4.4.1.2 Hardware Description

A. Computer

The central processing unit centers around a Digital Equipment Corporation VAX Computer with 512K to 1024K bytes of core storage. The VAX computer is suggested for several reasons. The fast execution speed and 32-bit word size of the VAX make it ideal for image processing applications. The reliability and ease of use of the VAX are such that it is becoming the industry standard.

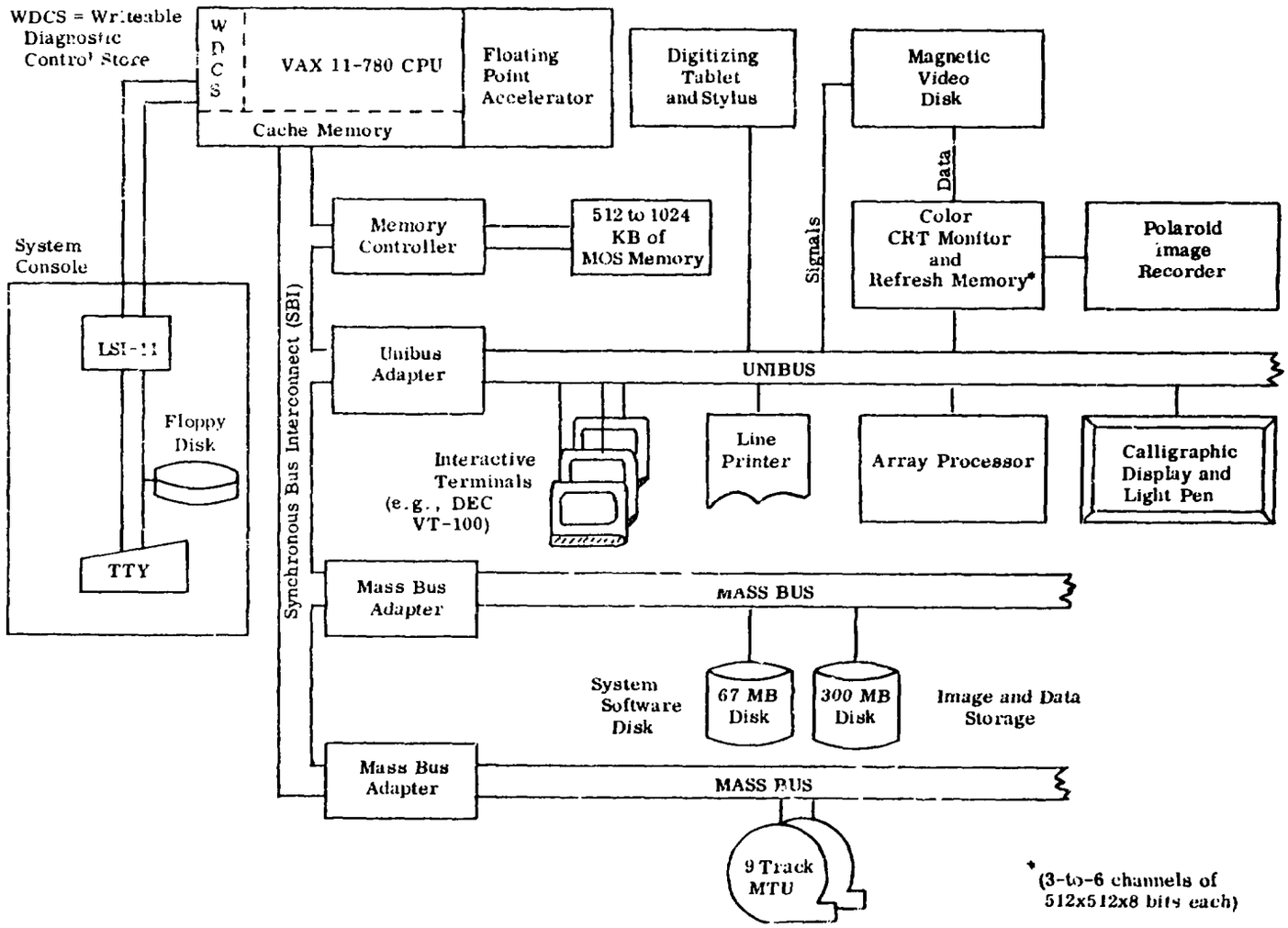


Figure 4.4.1-1. Advanced Image Synthesis System Hardware Block Diagram

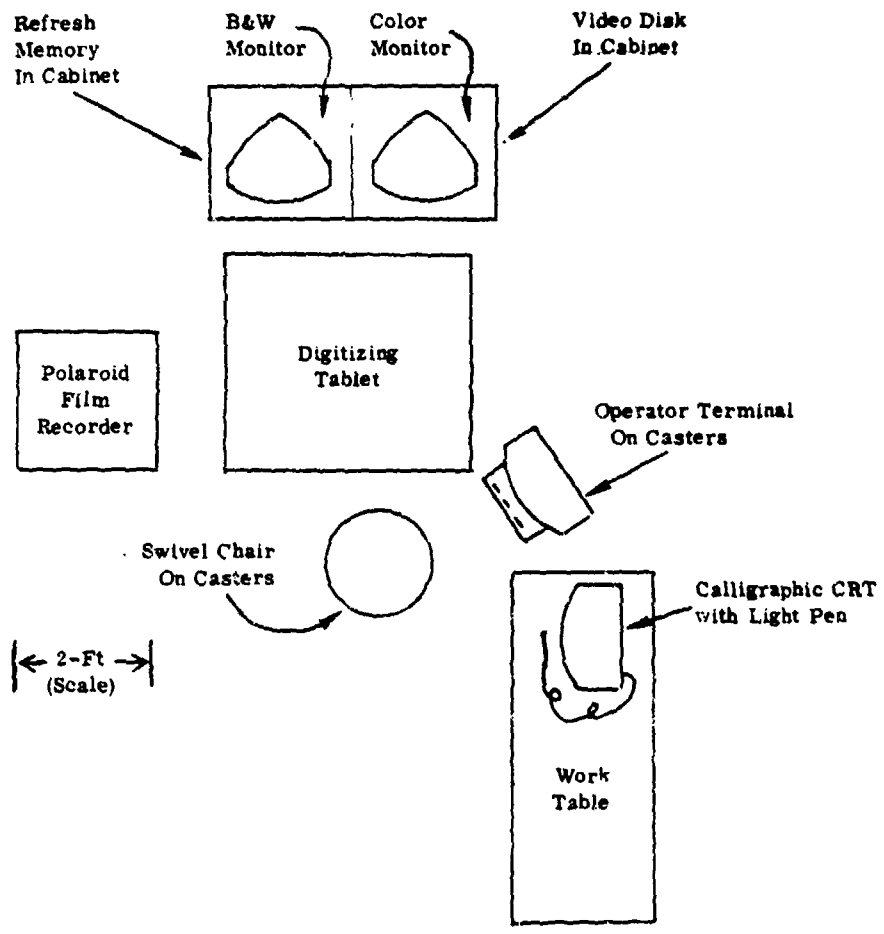


Figure 4.4.1-2. Floor Plan of Typical Graphic Work Station

The Main Memory Subsystem will contain one or two memory controllers. Each controller can handle from one to sixteen RAM arrays. Main memory arrays are MOS RAM integrated circuits with a cycle time of 600 nanoseconds. A memory controller can access a maximum of 4 million bytes.

B. Array Processor (optional)

A conventional general purpose computer must do a variety of tasks well. Individual computer programs must be organized to fit the architecture of the computer. With today's technology resulting in lower cost components, computer elements can be built which are dedicated to specific tasks. Most computer graphic and image processing algorithms are structured in vector/matrix form. An array processor is ideally suited for this type data structure. The use of array processors allows a task to be divided and distributed for maximum system throughput. One drawback of array processors is the difficulty of programming them (usually via assembly language).

General Electric has had direct experience with array processors manufactured by both Floating Point Systems, Inc., (AP-120B) and CSPI (MAP/300). These two processors are capable of 10-to-12 million floating point operations per second and are considered the industry standards.

C. Cathode Ray Tube (CRT) Monitors

Generated or stored images can be displayed on a 512-by-512-color monitor supported by a minimum of three channels of refresh memory (8 bits per pixel per channel). A second color or black and white monitor and one to three additional channels of refresh memory can be added if desired. This addition will permit the simultaneous display of two raster scan images.

The recommendation of a 512-by-512 pixel display over an ultra-high 1024-by-1024 pixel device is based on several factors. First, the software developed for the latter display will operate more slowly and require far more host memory than the former. The refresh memory would be at least four times greater in size. Secondly, video recording equipment for the 1024-by-1024 display is not yet available. Finally, it has been General Electric's experience that the lower resolution display is completely adequate for laboratory development purposes. Nonetheless, high resolution display systems are indeed commercially available as evidenced by

Table 4.4.1.2-1 which has been excerpted from a recent General Electric report (Appendix IV.3).

D. Polaroid Image Recorder (optional)

Polaroid image recorders will produce a 4-inch by 5-inch, or 8-inch by 10-inch picture of anything that is displayed on a CRT monitor. These devices are convenient for obtaining a quick high quality hard copy of a scene. The two primary manufacturers are the Dunn Instrument Company and Matrix Company. General Electric has experience with units manufactured by both companies. They are nearly identical in operation; the primary consideration in choosing between them may be that of cost.

E. Magnetic Video Disk (optional)

A magnetic video disk is a useful device for recording and storing animated sequences. It is the device used in sporting events for instant replay. The displayed sequences may, for example, represent simulated "snapshots" along a proposed route. The individual scenes stored on the disk may be played back one frame at a time, or in variable speed animation. Two currently available systems are those manufactured by Echo Science Corporation (Division of Arvin Company) and Eigen Video Company. These systems can record 10 to 20 seconds of video, which corresponds to 300 to 600 images. The Arvin-Echo disk can be played back in either forward or reverse modes.

F. Digitizing Tablet

A digitizing tablet is a flat rectangular table used for graphical input. For example, a map can be mounted on the surface of the tablet and the operator can then input data to the computer by tracing map features with a stylus. Digitizing tablets come in a wide range of styles, sizes, and prices.

G. Calligraphic CRT with Light Pen

Calligraphic CRT's are used to display images of the line drawing type. A light pen is an instrument that allows the user to point directly to an object on the screen, thus identifying it to the system. A light pen contains a switch that the user activates by pushing a button or pressing the tip of the pen against the CRT. This switch is used to indicate to the computer when the light pen is active.

Table 4.4.1.2-1. Video Frame Buffer Systems

	Approximate Price	Memory Configuration	Pixel Access	Color Map	Programmable Processor	Host(PDP-11) Interface	Desirable Facilities	1024 x 1024 Display Upgrade
DeAnza IP5000	$19,525	Restricted 8 or 12	1.8 Ms	4-4-4 (1024)	No	DMA Unibus	Weak	N/A
AED 512	$19,820	Modular 8 Max	1.0 Ms	8-8-8 (256)	No	DMA Unibus	Good	N/A
3 Rivers CVD	$20,000	RLE Format		5-5-5 (64)	No	Extended Memory	Weak	N/A
Grinell GMR-27	$22,000	Modular 32 Max	1.5 Ms	8-8-8 (1024)	No	DMA DR11-B	Weak	Rough
Grinell GMR-270	$23,200	Modular 32 Max	1.5 Ms	8-8-8 (3-256)	No	DMA DR11-B	Weak	Rough
ADI Light-50	$23,300	Modular 16 Max	1.2 Ms	5-6-5 (256)	Yes-download TMS9900	DMA like DR11-B	Good	N/A
Lexidata 3400	$25,800	Modular 16 Max	1.0 Ms	8-8-8 (256)	Difficult	DMA Unibus	Good	Trade-in
Genisco GCT-3000	$31,050	Modular 14 Max	1.5 Ms	8-8-8 (256)	Yes 2k RAM 6CT 3011	DMA Unibus	Good	Fair
DeAnza VC 5000	$34,950	Restricted 16 only	1.2 Ms	4-4-4 (1024)	Yes LSI-11	DMA via high speedport	Weak	Very Rough
Aydin 5216	$42,500	Modular 16 Max	1.0 Ms	8-8-8 (256)	1 Mega Word Intel 8086 (Forth)	DMA DR11-B	Excellent	Fair
Norpak VDP	$44,000	Modular 32 Max	1.5 Ms	8-8-8 (256)	Yes-download Bit-Slice Micro	DMA DR11-B	Very Good	N/A
DeAnza IP 5000	$48,000	Restricted 24 only	800 Ns	8-8-8 (3-256)	User Programmable Pipeline Array Proc.	Extended Memory	Fair	N/A
Ikonas	$48,000	Modular Max 40	400 Ns	8-8-8 (1024)	Fast 32 bit Micro	DMA DR11-B	Very Good (excellent Poss.	Built-in
Ramtek 9400	$53,100	Modular 16 Max	1.12 Ms	8-8-8 (1024)	Not Recommended Z-80	DMA DR11-B	Excellent	Rough
Comtal Vision One/20	240K to $700	Modular Max 512	1.5 Ms	8-8-8	LSI Micro	DMA DR11-B	Good	Rough

H. Color Image Scanner (optional)

Image scanners are high-speed digital microdensitometers. A system incorporates an electro-optical rotating drum which rapidly converts photometric or photogrammetric data on the film positives, negatives, or transparencies to digital form for computer processing. A unit for this purpose is currently available at Rome Air Development Center. An image scanner need not be in the same room as the rest of the graphic work station.

I. Floating Point Accelerator

The floating point accelerator is an optional high-speed processor extension. The accelerator enhances the performance of arithmetic operations.

J. Magnetic Disk Drives

Disk drives provide a reliable and high performance means of storing large amounts of data. All VAX disk drives use top-loading platters. Drives of different capabilities can be mixed on the same Mass Bus Controller.

K. Magnetic Tape Units

Each Magnetic Tape transport holds one 9-track reel with a capacity of 40 million characters per reel. The transport reads 45 inches per second with an average transfer time of 14 microseconds at 1600 bits per inch. Faster tape units can be purchased either from DEC or from other companies.

L. Line Printer

A Line Printer is treated as a spooled sharable device managed by multiple operator-controlled queues. A line printer can also be allocated to an individual program.

M. Interactive Terminals

Up to 96 interactive terminals can be connected to the VAX-11/78 system. The proposed system will probably only require two to four terminals.

N. Mass Bus

The processor interface for a Mass Bus peripheral is the Mass Bus Adapter. The Mass Bus Adapter performs control, arbitration, and buffering functions. Up to four Mass Bus Adapters can be placed on the SBI.

O. Unibus

All devices other than the high speed disk drives and magnetic tape transports are connected to the Unibus. The Unibus is connected to the SBI through the Unibus Adapter. The Unibus Adapter does priority arbitration among devices on the Unibus.

P. Video Interface

The video interface between the CRT's and the Polaroid image recorder will be a standard National Technical Standard Committee noncomposite analog video interface.

4.4.2 SUPPORT SOFTWARE ENVIRONMENT

The software support for the hardware system includes the VAX/VMS operating system, the VAX utility modules which it hosts, and graphics terminal and array processor support software. The VAX utilities include the EDT text editor, the VAX/VMS command Language Interpreter, the VAX-11 FORTRAN IV-PLUS compiler, the VAX-11 Linker, the VAX-11 MACRO Assembler, the VAX-11 Common Run Time Procedure Library, and the VAX-11 Symbolic Debugger. These allow the user to create, run, test, and modify software and manage data base storage and retrieval. Table 4.4.2-2 lists the DEC manuals which document this software.

The graphics terminal support software consists of the Tektronix Plot 10 Terminal Control System (TCS), a collection of Fortran subroutines which permit the user to generate, geometrically manipulate, and display calligraphic images under FORTRAN program control. In addition, the Tektronix Graphics Tablet Utility Routines (GTUR) provide for capturing graphic data from the digitizing tablet. Table 4.4.2-3 lists the manuals which document the graphics terminal support software as well as those which document the Floating Point Systems Array Processor support software.

4.4.3 INTERFACES

The interfaces between the components of the proposed hardware system are illustrated in Figure 4.4.1-1 and are described below.

Table 4.4.2-2. Manuals for DEC VAX Software System

Document Title	DEC Order Number
VAX-11 Disk Save and Compress User's Guide	AA-D739A-TF
VAX/VMS I/O User's Guide	AA-D028A-TE
and Update Notice No. 1	AD-D028A-T1
VAX/VMS Guide to Writing a Device Driver	AA-H499A-TE
Introduction to VAX-11 Record Management Services	AA-D024B-TE
VAX-11 Record Management Services Reference Manual	AA-D031B-TE
VAX-11 Record Management Services User's Guide	AA-D781B-TE
IAS/RSX-11M RMS-11 MACRO Programmer's Reference Manual	AA-0002A-TC
Introduction to RMS-11	AA-0001A-TC
RSX-11M RMS-11 V1.5 Utilities User's Guide	AA-D083A-TC
VAX-11 SORT User's Guide	AA-D113A-TE
PDP-11 SORT Reference Manual	AA-3341C-TC
VAX-11 FORTRAN IV-PLUS Language Reference Manual	AA-D034A-TE
VAX-11 FORTRAN IV-PLUS User's Guide	AA-D035A-TE
VAX/VMS Command Language User's Guide	AA-D023A-TE
and Update Notice No. 1	AD-D023A-T1
VAX-11 Linker Reference Manual	AA-D019A-TE
VAX-11 Symbolic Debugger Reference Manual	AA-D026A-TE
VAX-11/RSX-11M Programmer's Reference Manual	AA-D020A-TE
VAX-11/RSX-11M User's Guide	AA-D037A-TE
VAX-11 MACRO Language Reference Manual	AA-D032B-TE
VAX-11 MACRO User's Guide	AA-D033B-TE
VAX-11 Common Run-Time Procedure Library Reference Manual	AA-D036A-TE
VAX-11 Guide to Creating Modular Library Procedures	AA-H500A-TE
VAX-11 Text Editing Reference Manual	AA-D029A-TE

Synchronous Bus Interface (SBI)

The SBI is VAX's internal backplane. It is the bus that conveys addresses, data, and control information between CPU, main memory, and peripheral devices. The SBI has a cycle time of 200 nanoseconds and can transfer 32 bits each cycle. The maximum SBI transfer rate is 13.3 million bytes per second.

Arbitration of the SBI is distributed. Every unit on the SBI has its own arbitration line. Arbitration lines are ordered by priority. Every unit monitors all the arbitration lines in each cycle to determine if it will get the next cycle.

4.4.4 LABORATORY FACILITY CONSIDERATIONS

It is suggested that the proposed system be installed in a room approximately 18 feet long, by 24 feet wide, by 10 feet high. The room should have a raised floor so that interface and power cables can be secured out of the system operators' way. It is suggested that the floor, ceiling, and possibly walls be covered with sound absorbing material. Illumination should be via white light to allow accurate color comparison to be made.

The proposed VAX-based computer system and the other equipments are designed to perform under a fairly limited range of environmental conditions. Specifically, VAX systems operate best within the following limits.

Temperature:	Air-conditioned room, 15°C (59°F) to 32°C (90°F); (Optimum temperature for personnel and equipment is 70°F ± 5°).
Relative Humidity:	20% to 80% with maximum wet bulb 25°C (77°F) and minimum dew point 2°C (36°F).
Altitude:	Up to 2400 meters (8000 feet) above sea level.

Table 4.4.2-3. Graphics and Array Processor Software Manuals

Vendor	Publication	Part Number
Tektronix, Inc.	Plot 10 Terminal Control System (TCS) Manual	070-2242-00
"	Plot 10 TCS User Manual	070-2241-00
"	Plot 10 Graphics Tablet Utility Routines User Manual	070-2253-00
Floating Point Systems, Inc.	Release 2 Software Information	860-7370-000

4.5 COST FACTORS

It is assumed that a medium performance 32-bit computer will be available and will not need to be procured specifically for the implementation of the hardware design proposed here. Therefore, only the costs of the hardware components of the proposed graphics work station are identified. The costs of the proposed software modules are subject to negotiation and will not be covered here.

Hardware Item	Cost	Discussion
Digitizing Tablet and Stylus	$3,000 to $10,000	Cost is related to the size of the bed of the table.
Refresh Memory and CRT Displays	$10,000 to $50,000	Cost is related to the number of memory planes, number and size of CRT's, and whether color or monochrome. (A Comtal system is available at RADC and may be used for this purpose.)
Calligraphic Display and Light Pen	$5,000 to $10,000	
Polaroid Image Recorder (optional)	$12,000 to $18,000	
Magnetic Video Disk Unit (optional)	$25,000 to $50,000	
Image Scanner (optional)	$50,000 to $100,000	(An image scanner is available at RADC.)

SECTION 5

CONCLUSIONS AND RECOMMENDATIONS

5.1 INTRODUCTION

The principal object of this research was to study efficient methods of representation, storage, and display of sensor scene simulation, particularly for low altitude terrain views. Carrying out the tasks described in Section 1, paragraph 1.3 (APPROACH), led to the conclusions and recommendations organized into the following areas below.

5.2 SCENE CONTENT REQUIREMENTS

There have been many successful studies which have provided useful information on perceptual aspects of scene detail requirements. Examples include experiments which determined the minimum number of raster lines required for target acquisition, and measurements of perception of images with reduced gray scale ranges. There is relatively less information available on more global requirements for total mission planning. Conceptually, such requirements are at a semantically higher level and thus inherently more difficult to assess. Therefore, in this study a thorough analysis of total scene content requirements was carried out for some rather large geographic areas. Since the analysis was based on overall mission needs for low level visual flight simulation, its scope was more comprehensive than solely perceptual studies. Results of that analysis include the recommendation that all the graphic information on large scale maps be incorporated into scene data bases as a minimum. Additional visual detail must be added to provide both motion cues and correct environment perception. This detail must be provided by the introduction of random 3-D models and visual texture on planar surfaces in order to achieve the goals of target identification, threat appraisal, navigation, and weapons delivery planning. Texture is an important enhancement which efficiently provides depth, orientation, and relative position cues on large polygons without introducing extra edges. Quantitative analysis led to the conclusion that texture can provide cues which would require between five times and several hundred times as many edges in the absence of texture, depending on the visual surface being modeled.

Scene data structure and geometric content can be nearly identical for simulation of several different sensor types; thus the same data base and processing algorithms can be used for several sensors. However, parameters and processing details will differ for different sensors. For example, perspective display of the same geometric

shape yields both a FLIR and visual image of a vehicle, but the color lookup table and vehicle segmentation may differ because of the physical differences between optical and thermal emission.

Low level flight is characterized by extreme perspective effects. That is, objects in the foreground loom much larger than more distant objects, and hidden object calculation burdens are larger than for high level flight. Thus, multiple level of detail representation of the same scene is essential, and proper linking between levels is necessary in the data structure in order to provide visual richness in the foreground while preventing processing overload at extreme ranges in the same scene.

5.3 TECHNOLOGY REVIEW

Sensor simulation is an application of computer graphics with emphasis on characteristics and requirements outside the mainstream of academic computer graphics. For example, sensor simulation scenes consist primarily of terrain rather than the isolated discrete objects found in most computer graphics images. The quantity of data to be processed in sensor simulation is thus larger and processing involves more massive hidden surface calculations than most graphics applications because the attendant polygon sorting costs rise faster than the number of polygons. Thus, the technology review placed heavy emphasis on hidden surface algorithms because of the central role they play in efficiently generating sensor scenes. In contrast, special effects in computer graphics such as translucency and surface gloss were not emphasized because they are not of importance to sensor simulation. Spline theory, another currently popular topic in computer graphics, introduces processing inefficiency without commensurate improvement of sensor scene simulation quality.

A number of hidden surface algorithms were analyzed and compared. The analysis revealed that special topological characteristics of the geometric representation of terrain can be exploited to improve scene simulation efficiency by greatly reducing the complexity of hidden surface calculation. Principal among these characteristics is the fact that terrain polygons represent a tessellated sheet or surface. Hence, distance to centroid of each polygon can be used to determine visibility priority without resorting to more complex methods required for less constrained arrangements of polygons. Addition of planimetric data to terrain poses no additional visibility calculation problems because new faces are subdivisions of old ones, with a simple layering priority used for faces in the same plane. The addition of 3-D objects to terrain, however, does impose a larger processing burden. Thus study recommends that such burdens be alleviated by using methods developed by General Electric for flight simulators. In this approach, objects are assigned the priority

of the terrain face with which they are associated so that separate priority calculations need not be carried out for objects on distinct faces. Objects on the same face can be hierarchically divided into subgroups whose constituents are assigned precalculated view-independent priorities. Thus, much of the hidden surface calculation is carried out during data base definition, increasing the efficiency of the scene generation process.

Aside from priority calculations, another major computation burden in sensor scene simulation is imposed by raster scan conversion, the process of filling pixels with the color corresponding to the projected image of the data base onto the screen. There are two basic approaches to this problem. The first involves filling the pixels sequentially a row at a time; this is called scan-line processing. The second approach processes random screen position in the order determined by the data base, i.e., a polygon at a time. Efficiencies in the first approach stem from the fact that each pixel is visited only once and that polygon rendering exploits scan-line coherence, the property that intersections of a polygon edge with successive scan lines can be computed incrementally rather than by solving the linear equation anew. The computational efficiency realized by avoiding such calculations can be massive in scenes containing thousands of edges, a common situation in sensor scenes. The efficiency of scan-line order algorithms has been borne out by their extensive use in CIG systems. Their advantage there lies in the fact that each pixel value may be computed on the fly and sent directly to video display, avoiding the expense of a video refresh memory. Unfortunately, this also puts severe demands on processing speed, and leads to overload situations in low level flight because so many polygons pile up on the horizon over a narrow range of scan lines. In nonreal-time scene simulation, a video refresh memory is required anyway, so the only saving comes about from the incremental character of scan-line calculations.

The use of CIG systems in flight simulation has provided a crucible for weeding out inefficient algorithms from the gamut of those found in non-realtime computer graphics. Thus, the technical review of CIG systems in this study led to some interesting conclusions and recommendations for sensor simulation. Calligraphic systems, though efficient from the point of view of real-time hardware, do not provide enough image tone or color richness to convey continuous levels of shading required for LLLTV or daytime visual scene depiction; thus raster scan displays are preferable. True raster scan display is also recommended because many of the sensors are raster scan devices; thus, images appear more natural on raster displays than on calligraphic displays. Recently some new CIG algorithms have been developed which take advantage of decreasing chip cost and increasing chip density due to VLSI advances. These algorithms use random access to frame buffers rather than scan-line coherence, rearranging the order of computations. The result is that far fewer polygons need to be processed and computations need only be carried

out to the accuracy of display resolution. Furthermore, there are no per-scan line overloads in such an approach; this is particularly important for horizon processing in low level flight. While more memory is required for such methods compared to scan-line processing, decreasing chip costs move the cost-effectiveness crossover point in the direction of this approach. The Evans and Sutherland CT-5 span processing system is a good example of a realtime system using this approach; processing efficiencies also apply to software for non-realtime sensor scene simulation. For example, a scan-line approach would have to process all faces which project onto a visual horizon whereas a span processing approach emulating CT-5 type operations only processes those large faces in the foreground which occlude those behind.

The review of image processing led to the conclusion that it only plays a minor role in the simulation of sensor scenes although it may play a larger role in the derivation of scene data bases from photographic and sensor data. Higher level functions of image understanding suffer from the difficulties encountered in all areas of artificial intelligence, namely, lack of generalization, inflexibility, and inability to autonomously process complex real world data. There is a promising possible new application for image enhancement in sensor scene generation to highlight visual features of special interest.

5.4 SYSTEM DESIGN

Analysis of the tasks to be carried out by a laboratory system for research in sensor simulation led to the conclusion that 16-bit minicomputers are too small, too slow, and do not carry enough precision in their arithmetic operations to carry out the geometric computations characteristic of scene generation. Large computer systems were judged inappropriate by reason of cost and design. The inflexibility of CIG systems limits their usefulness as research tools; their great expense, caused by real-time performance requirements, represents a tremendous penalty on scene generation efficiency, especially in a research environment where real-time images are not required. Performance and efficiency considerations lead to the recommendation that the research system be structured around a mid-size 32 bit computer configured as follows:

Computer Subsystem

- 1 - Medium-size, 32-bit computer system (with 512 kilobytes to 1 megabyte of core storage)
- 1 - Array processor (optional)
- 1 - Line printer

Graphic Workstation

- 1 - Digitizing tablet and stylus
- 1 - Color CRT monitor (with 3-to-6 channels of refresh memory)
- 1 - Black and white CRT monitor (with refresh memory)

Computer Subsystem (Continued)

- 2 - Disk drives (one 67 mb and one 300 mb)
- 2 - Nine-track magnetic tape units
- 1 - System console
- Up to 3 interactive terminals

Graphic Workstation (Continued)

- 1 - Calligraphic display and light pen
- 1 - Polaroid film image recorder (optional)
- 1 - Magnetic video disk (optional)
- 1 - Color image scanner/digitizer (optional)

Scene generation software should be implemented in a high-level language, following good software design principles including modularity, structure, and liberal use of comments. Standard utility software, including editors, compilers, and math libraries, should be included with the system.

5.5 DIRECT DISPLAY OF GRID DATA BASES

One of the most significant conclusions of this study is that the direct display of grid data bases will replace current polygon data base display methods in the near future because of recent advances in VLSI and parallel algorithm design. Direct display of grid data bases has been neglected relative to polygon methods for a variety of reasons. Among these are that more memory and processing are required for grid data base display, and that geometric problems related to sampling mismatches, perspective calculation, and hidden surface calculation give rise to visual artifacts which are often reduced by ad hoc methods. On the positive side, grid data base display methods require no preprocessing of data to yield polygons and no level of detail linkage. Furthermore, the images are strikingly more realistic than polygon network representations of terrain. The extra memory requirements for grid display become less significant as VLSI memory chips become denser and less expensive. A breakthrough in grid data base algorithm design was developed during this study. Though not yet implemented, it suggests valid solutions to geometric problems in grid data base display and lends itself naturally to parallel processing in VLSI hardware. The following paragraph describes principal characteristics of the algorithm.

The new algorithm accesses rows and columns of data independently. It completely decouples depth-related calculations (haze, occulting and perspective) from scan-line related calculations (azimuth ray images). Such properties encourage parallelism in the design of compact, inexpensive hardware systems for generating realistic, geometrically valid perspective images of terrain directly from Defense Mapping Agency (DMA) data. These properties also provide for straightforward

enhancement of close range scenes by elevation interpolation and the interleaving of models or special features during either data definition or display generation. The grid data structure is ideal for fractal interpolation to generate realistic terrain images at close ranges without storing the corresponding elevation data. Anti-aliasing, valid occlusion of invisible features (priority) and distance fading (haze) calculations are inherent in the algorithm. Computational organization exploits highly parallel hardware efficiency by making row and column calculations simple, local and independent. Trigonometric functions are avoided entirely. The efficiency of the algorithm depends on the fact that the grid coordinate system is aligned with the viewray through the viewpoint perpendicular to the viewscreen. This results in constant range for all grid points in a row parallel to the screen, which greatly simplifies data access and computation.

A significant recommendation of this study is that algorithms for direct display of grid data bases be intensively studied in future research. Recent hardware advances and new algorithms suggest that these methods will soon prove more efficient at generating sensor simulation scenes of much higher quality than polygon based methods. New areas which should be pursued include enhancement of scenes by modification of elevation grids and superimposition of reconnaissance photography on planimetric data. Future studies should also investigate implementation of parallel algorithms in software and hardware to take advantage of new design economies resulting from recent advances in off-the-shelf VLSI devices.

I. GLOSSARY

Mnemonic	Definition
AWS	Advanced Weapon System
B&W	Black and White
C^3I	Command, control communications and intelligence
CCT	Computer-compatible tape
CIG	Computer image generation
CPU	Central processor unit
CRT	Cathode ray tube
DEC	Digital Equipment Corporation
DMA	(United States) Defense Mapping Agency
FLIR	Forward looking infrared
GE	General Electric Company
kb	Kilobyte (one thousand bytes)
LLLTV	Low light level television
mgb	Megabyte (one million bytes)
MOS	Metal oxide semiconductor
MTU	Magnetic tape unit
pixel	Picture element
RADC	Rome Air Development Center, Rome, NY
RAM	Random access memory
SBI	Standard bus interface
TTY	Teletypewriter
VAX	Digital Equipment Corporation VAX 11 Computer
WDCS	Writeable diagnostic control store

II. TOPICAL BIBLIOGRAPHY

KEY TO TOPICS:

C- CIG AND SIMULATION
D- GRID DATA BASE DISPLAY AND GENERATION
F- FLIR, LLLTV, MICROWAVE AND OTHER VISUAL SENSOR DISPLAYS
G- COMPUTER GRAPHICS
P- PERCEPTUAL PSYCHOLOGY IN VISUAL TRAINING
V- ARITIFICIAL VISION AND IMAGE PROCESSING

C- CIG AND VISUAL SIMULATION

C.1 Beardsley, H., Bunker, W. M., Eibeck, A., Juhlin, J., Kelly, W., Page, J., and Shaffer, L., Advanced Simulation in Undergraduate Pilot Training: Computer Image Generation, General Electric Company, Daytona Beach, Florida for Air Force Human Resources Lab, WPAFB, Ohio, AFHRL-TR-75-59(V), November 1975.

C.2 Bunker, M., "The Simulation Data Base - Describing the World to the Computer", Proceedings of the Summer Computer Simulation Conference, Houston, Texas, July 8-11, 1974.

C.3 Dichter, W., Doris, K., and Conkling, C., "A New Approach to CGI Systems", Gould, Inc., in Proceedings of the Second Interservice/Industry Training Equipment Conference, Salt Lake City, November 1980, pp. 102-109.

C.4 Doenges, P. K. , Introduction to Computer Image Generation Technology, Flight Simulation short course presented at Dayton, Ohio , March, 1980.

C.5 Doty, A. B. and Hoog, T. W., Components of a Simulator, Course notes, University of Dayton AIAA Flight Simulation Short Course, Dayton, Ohio, March, 1980.

C.6 Forbes, J. E., "The Stress is Now Toward Better Training in Less Space at Lower Cost", ICAO Bulletin, April 1978, pp. 22-25.

C.7 General Electric, Airborne Electro-Optical Sensor Simulation, Final Report prepared for AFHRL, Brooks AFB, Texas, AFHRL-TR-75-35, 1975.

C.8 General Electric, Computer Display of Height Fields, G. E. Ground Systems Dept, Report No. 79ASD006, April, 1979.

C.9 General Electric, Instruction Manual for Visual Three-View Space-Flight Simulator, Prepared for NASA Manned Spacecraft Center, Houston Texas under contract NAS 9-1375, Advanced Electronics Center, General Electric, Ithaca, New York.

C.10 General Electric, Instruction Manual for Digital Contact Analog, Volume A: theory, Operation, and Maintainance, Prepared for Joint Army Navy Instrumentation Research Program, Contract No. NR 4057(00), by Electronics Lab, General Electric, Syracuse, New York, May 1966.

C.11 General Electric, Surface Map Texture for Computer Generated Images, G. E. Ground Systems Dept, Report No. 76ASD013, December, 1976.

C.12 General Electric, Tactical Situation Display System, G. E. Space Division IR&D Report Numbers 77ASD009, 78ASD024, and 79ASD009, 1977/1978/1979 respectively.

C.13 General Electric, Techniques for Displaying Digital Topographic Data, G.E. Ground Systems Dept, Report No. 79ASD009, July, 1979.

C.14 General Electric, Transformation Program for Radar and Image Simulation, G. E. Ground Systems Dept, Report No. 78ASD010, August, 1978.

C.15 Goldstein, R. A., and Nagel, R., "3-D Visual Simulation", Simulation, pp. 25-30, 1971.

C.16 Handberg, G. O., "Advanced CGI Visual Technology Reshapes Pilot Training Possibilities", ICAO Bulletin, April 1977, pp. 27-32.

C.17 Johnson, J. C., Physical Meteorology, MIT Press, Cambridge, Mass., 1954.

C.18 Middleton, W. E. K., Visibility in Meteorology, University of Toronto Press, Toronto, 1941.

C.19 Ranjbaran, S. E., and Swallow, R. J., "COMPUTROL in Flight Simulation", in Proceedings of Eurographics '80 Conference, C. E., Vandoni, Editor, North Holland Publishing Co., 1980, pp. 321-329.

C.20 Rowley, T. (Marconi Radar, Ltd), "Computer Generated Imagery for Training Simulators", pp. 223-232 in Proceedings of Computer Graphics Conference, Brighton, England, August 1980.

C.21 Schachter, B. J. and Ahuja, M., "A History of Visual Flight Simulation", Computer Graphics World, Vol. 3, May 1980, pp. 16-32.

C.22 Schachter, B. J., "Real Time Display of Texture", in Proceedings of the Fifth International Conference on Pattern Recognition, Miami, Fla., December 1980, pp. 789-791.

C.23 Schachter, B., "Computer Generation of Full Colored Textured Terrain Images", Proceedings of the 1(st) Interservice/Industry Training Equipment Conference, Orlando, Florida, November 1979, pp. 367-374.

C.24 Schachter, B., Computer Program Product Specification Mathematical Model for Transformation Program, C-130 Visual System, Contract F33657-78-C-0421, General Electric Co., Daytona Beach Florida, August, 1978.

C.25 Schnitzer, A. P., "A Data Base System for Digital Image Generation", Proceedings on the 9th Industry Conference, Naval Training Equipment Center, Orlando Florida, Nov. 1976, pp. 103-113.

C.26 Schumacker, R. S., "A New Visual System Architecture", in Proceedings of the Second Interservice/Industry Training Equipment Conference, Salt Lake City, Utah, 1980, pp. 94-101.

C.27 Shahat, M., "Esoteric Simulator Pact to Hughes - The F-18 Weapons Tactics Trainer", Military Electronics/Countermeasures, Nov. 1979, pp. 80-81.

C.28 Smare, A. D., "West Yorkshire Traffic Simulator", in Proceedings of Computer Graphics Conference, Brighton, England, August 1980.

C.29 Specification for AH-64 Combat Mission Simulator, Device 2B40, No. 222-1162, Naval Training Equipment Center, 1, June, 1978.

C.30 Swallow, R., Goodwin, R., Draudin, R., "COMPUTROL: A New Approach to Computer Generated Imagery", SPIE Vol 162: Visual Simulation and Image Realism, 1978, 16-25.

C.31 Tanner, P. P., "Dynamic Illustrative Graphics for Simulation" in Proceedings of Computer Graphics Conference, Brighton, England, August 1980.

C.32 Vorst, C. J., "A New Visual Simulation Technique for Pilot Training", in Proceedings of the Ninth Industry Conference, Naval Training Equipment Center, Orlando Florida, Nov. 1976, pp. 115-125.

D- GRID DATA BASE DISPLAY AND GENERATION

D.1 Bunker, W. M., and Heartz, R., Perspective Display Simulation of Terrain", General Electric Company, Daytona Beach, Florida study for Air Force Human Resources Lab, AFHRL-TR-76-39, Contract AFHRL F33615-75-C-5243, June, 1976.

D.2 Defense Mapping Agency, Product Specification for Digital Landmass Systems Data Base, Defense Mapping Agency, St. Louis, Missouri, July, 1977.

D.3 Dungan, W. Jr., "A Terrain and Cloud Image Generation Model", SIGGRAPH '79 Proceedings, 1979, pp. 143-150.

D.4 Gillespie, A. R., and Kahle, A. B., "Construction and Interpretation of a Digital Thermal Inertia Image", Photogrammetric Engineering and Remote Sensing, Vol 43, August 1977, pp. 983-1000.

D.5 Heartz, R. A., "Terrain Display Simulation for Mission Briefing", Proceedings of the 11(th) Annual Simulation Symposium, March, 1978.

D.6 Hendley, C. D., and Hect, S., "The Colors of Natural Terrain and Their Relation to Visual Color Deficiencies", Journal of the Optical Society of America, Vol 39, 1949, pp. 870-873.

D.7 Horn, B. K. P., "Hill Shading and the Reflectance Map", Proceedings of the IEEE, Vol. 69, 1981, pp. 14-47.

D.8 Horn, B. K. P., Image Intensity Understanding, MIT Artificial Intelligence Lab Memo AIM 335, Cambridge, Mass, August, 1975.

D.9 Jancaitis, J. R., Modelling and Contouring Irregular Surfaces Subject to Constraints, Report Number ESS-3325-101-75, University of Virginia, Dept. of Engineering Sciences and Systems, January, 1975.

D.10 Krinov, E. L., Spektral'naia Ostrazhatel'nia Sposbnost' Prirodynykh Obrazovanii (Special Reflectance Properties of Natural Formations), Lab. Aerometodov, Akad. Nauk SSSP, Moscow, 1947.

D.11 Mark, D. M., "Computer Analysis of Topography: a Comparison of Terrain Storage Methods", Geografiska Annaler, Vols. 3-4, Series A, 1975, pp. 179-188.

D.12 Mark, D. M., "Phenomenon Based Data Structuring and Digital Terrain Modelling", Geo-Processing, Vol 1, 1979, pp. 27-36.

D.13 Maxwell, J. R., Statistical Analysis of Selected Terrain and Water Background Measurement Data, Environmental Research Institute of Michigan, Ann Arbor, Michigan, July 1978.

D.14 McKeown, D., "Representation for Image Data Bases", in Proceedings of the Image Understanding Workshop in Cambridge Mass, May 1978, L. S. Bauman (Ed.), Scientific Applications Inc, SAI-79-749-WA, Arlington, Va, 1979, pp. 109-111.

D.15 Nickerson, D., Kelly, K. L., and Stultz, K. F., "Colors of Soils", Journal of the Optical Society of America, Volume 35, 1945, pp. 297-300.

D.16 Pendleton, R. L., and Nickerson, D., "Soil Colors and Special Munsell Color Charts", Soil Science, Vol 71, 1951, 35-43.

D.17 Pennderff, R., Luminous and Spectral Reflectances as Well as Colors of Natural Objects, Air Force Cambridge Research Center, Bedford Mass., Geographical Research Paper 44, 9ASTIZ Number AD-98766, AFCRS-TR-56-203, 1956.

D.18 Rounds, E., and Zavodny, T., Dynamic Guided Missile Retargeting Study, Technology Service Corporation Report for Rome Air Development Center, RADC-TR-79-332, January, 1980.

D.19 Spooner, A. M., Breglia, D. R., and Patz, B. W., "REALSCAN - A CIG System with Greatly Enhanced Image Detail", Interservice/ Industry Training Equipment Conference, 1980.

D.20 Strat, T. M., "Shaded Perspective Images of Terrain", MIT AI Lab Memo 483, March, 1978.

D.21 Weiman, C. F. R., "Continuous Anti-Aliased Rotation and Zoom of Raster Images", SIGGRAPH '80 Conference Proceedings", July 1980, pp. 286-293.

F- FLIR, LLLTV, MICROWAVE AND OTHER VISUAL SENSOR DISPLAYS

F.1 Bell, J. W., Bottik, I. P., and Lucero, A. B., Simulation Techniques for Airborne Electro-Optical Imaging Systems, Technology Services Corporation for AFHRL, AFHRL-TR-75-43, 1975.

F.2 Bunker, W. M., "Computer Generated Images Simulate Infrared and LLLTV Displays", Proceedings of the AIAA Visual and Motion Simulation Conference, Dayton, Ohio, April 1976, pp. 120-132.

F.3 Bunker, W. M., "Techniques for Simulation of Radar Displays", Proceedings of the Sixth Annual Simulation Symposium, Tampa, Florida, March 1973, pp. 1-25.

F.4 Bunker, W. M., and Heeschen, R., Airborne Electro-Optical Sensor Simulation, General Electric study for Air Force Human Resources Lab, AFHRL-TR-75-35, July 1975.

F.5 Bunker, W. M., "Computer Simulation of Electro-Optical Viewing Systems", Journal of Aircraft, April 1977, pp. 394-400.

F.6 de Groot, S., Human Factors Aspects of LLLTV and FLIR sensor Displays, Air Force Office of Scientific Research, AFOSR-TR-78-1237, 1978.

F.7 Dessau, H., AWS Image Simulation Study, RADC-TR-78-176, August, 1978.

F.8 Faintich, M. B., "Digital Sensor Simulation, 1980-1985", in Proceedings of the 13(th) Annual Simulation Symposium, Tampa, Florida, March 1980, pp. 181-190.

F.9 Heartz, R. A., and Bunker, W. M., "Radar Display Simulation for Training", Analog/Hybrid Computer Education Society Transactions, Vol. 3, June 1971, pp. 109-120.

F.10 Moldenato, E. D., Forward Looking Infrared Simulation, Naval Training Equipment Center, NAVTRAEQUIPCEN TN-39, Orlando, Florida, July 1974.

F.11 Moore, R. P., Hooper, J. O., and Hooper, E. T., "Interpretation of Microwave Radiometric Images", Proceedings of SPIE, Aerial Reconnaissance Systems, Vol. 79, 1976, pp. 146-153.

F.12 Neitman, R. L., and Kemper, M. R., Seasonal Sensor Scenics Handbook, Mead Technology Laboratories study for AFWRL contract F33615-76-C-1113, July 1, 1977.

F.13 Schult, N. L., and Wenthen, F., Advanced Weapon Systems (AWS) Image Simulation II Study, General Electric Co. study for Rome Air Development Center, RADC-TR-80-185, June 1980.

F.14 Scott, F., Hollands, P. A., and Hartabedian, A., "The Information Value of Sampled Images as a Function of the Number of Scan Lines per Object", Photographic Science Engineering, Vol 14, 1970, pp. 21-27.

F.15 Stenger, A. J., Stone, W. R., Berry, L., and Murray, J. T., Sensor Image Prediction Techniques, Technology Service Corporation Final Technical Report for Rome Air Development Center Contract F30602-79-C-0054, RADC-TR-80-388, February, 1981.

F.16 Stathacoupolos, A. D. and Gilmore, H. F., Selection of Mathematical Models of Target Acquisition by Electro-Optical Systems, Science and Corporation for Naval Weapons Center, Technical Publication 5928, June, 1977.

F.17 Tryon, M. E., VanDeman, G. L., Hogue, C. E., and Bridges, W. J., Computer Assisted Simulation of Electromagnetic Sensors, Naval Avionics Facility, TR 2021, September, 1975.

G- COMPUTER GRAPHICS

G.1 Baer, A., Eastman, C., and Henrion, M., A Survey of Geometric Modelling, Carnegie-Mellon University Institute of Physical Planning, Report No. 66, March, 1977.

G.2 Boutnight, W. J., An Improved Procedure for Generation of Half-Tone Computer Graphics Representations, University of Illinois Coordinated Sciences Lab., R-432, Sept. 1969.

G.3 Boutnight, W. J., "A Procedure for Generation of Three-Dimensional Half-Toned Computer Graphics Representations", Communications of the ACM, Vol 13, September, 1970, pp. 527-536.

G.4 Braccini, C., and Marino, G., "Fast Geometrical Manipulations of Digital Images", Computer Graphics and Image Processing, Vol. 13, 1980, pp. 127-141.

G.5 Carpenter, L. C., "Computer Rendering of Fractal Curves and Surfaces", Computer Graphics, July 1980, (SIGGRAPH '80 ISSUE).

G.6 Catmull, E., and Smith, A. R., "3-D Transformations of Images in Scanline Order", SIGGRAPH '80 Conference Proceedings, July 1980, pp. 279-285.

G.7 Chaikin, G. M., "An Algorithm for High Speed Curve Generation", Computer Graphics and Image Processing, Vol. 3, 1974, pp. 346-349

G.8 Chasen, S. H., Geometric Principles and Procedures for Computer Graphic Applications, Prentice-Hall, Englewood Cliffs, New Jersey, 1978.

G.9 Clark, J., "A VLSI Geometry Processor for Graphics", Computer, July, 1980.

G.10 Christiansen, H., and Stephenson, M., MOVIE.BYU - A General Purpose Computer Graphics Display System, Dept. of Civil Engineering, Brigham Young University, July 1978.

G.11 Christiansen, H., and Stephenson, M., Graphics Utah Style - 79: Notes from Workshop on Interactive Computer Graphics with Emphasis on the MOVIE system, Salt Lake City, Utah, June 25-29, 1979.

G.12 Eastman, C. M., "The Concise Structuring of Geometric Data for Computer Aided Design", pp. 31-58 in Data Structures, A. Klinger, K. S. Fu and T. L. Kunii (Eds.), Academic Press, 1977.

G.13 Elliot, R. L., "Computer Graphics in a Multiple System Multiple Device Environment", in Proceedings of Computer Graphics Conference, Brighton, England, August 1980.

Q.14 Encarnacao, J., Survey of and New Solutions for the Hidden-Line Problem, in Proceedings of Graphics Computing Symposium, Delft, Holland, October, 1970.

Q.15 Fischler, M. A. (Ed.), Interactive Aids for Cartographic and Photo Interpretation, SRI Project 5300.

Q.16 Foster, R. A., "Two Systems for Displaying the Results of Oil Reservoir Simulations", in Proceedings of Computer Graphics Conference, Brighton, England, August 1980.

Q.17 Frazer, J., Frazer, J., and Frazer, P., "Intelligent Three-Dimensional Modelling Systems", in Proceedings of Computer Graphics Conference, Brighton, England, August 1980

Q.18 Fournier, A. and Fussel, D., "Stochastic Modelling in Computer Graphics", Computer Graphics, July 1980, (SIGGRAPH '80 ISSUE).

Q.19 Giloi, W. K., Interactive Computer Graphics, Prentice-Hall, Englewood Cliffs, New Jersey, 1978.

Q.20 Hamlin, G., and Gear, C. W., Raster Scan Hidden Surface Algorithm Techniques, ACM Computer Graphics, Vol 11, Summer, 1977.

Q.21 Haralick, R. M., "Using Perspective Transformations in Scene Analysis", Computer Graphics and Image Processing, Vol. 13, 1980, pp. 191-221.

Q.22 Henderson, R. L., Geometric Reference Preparation Interim Report Two: The Broken Segment Matcher, Control Data Corp. Report for RADC, RADC-TR-79-80, April, 1979.

Q.23 Mathematical Applications Group, SYNTHAVISION: The Exciting New Way for You to CREATE DIMENSIONAL MOVING PICTURES IN COLOR, MAGI, Elmsford, New York.

Q.24 Mead, C. A., and Conway, L., Introduction to VLSI Systems, Addison-Wesley, Reading, Mass., 1980.

Q.25 Nelson, C. L., Henderson, R. L., Panton, D. J., Grosch, C. B., and Miller, W. J., Geometric Surface Shell Studies, Control Data Corp study for Rome Air Development Center, RADC-TR-80-65, March 1980.

Q.26 Newman, W. M., and Sproull, R. F., Principles of Interactive Computer Graphics, McGraw-Hill, New York, 1979.

Q.27 Ranjbaran, S. E., "A Mathematical Model for the Visual World", in Proceedings of the 11(th) Annual Conference on Modeling and Simulation, Instrumentation Society of America, Research Triangle Park, North Carolina, May, 1980, pp. 135-139.

Q.28 Romney, G. W., Computer Assisted Assembly and Rendering of Solids, University of Utah Dept of Computer Science, TR-4-20, 1970.

Q.29 Rothstein, J. and Weiman, C. F. R., "Parallel and Sequential Specification of a Context Sensitive Language for Straight Lines on Grids", Computer Graphics and Image Processing, Vol. 5, 1976, pp. 106-124.

Q.30 Russell, D. M., Constraint Networks, University of Rochester, Computer Science Tech Report TR-38., August, 1978.

Q.31 Schachter, B. J., "Decomposition of Polygons into Convex Sets", IEEE Transactions on Computers, Vol. C-72, Nov, 1978, pp. 1078-1082.

Q.32 Schachter, B. J., Rosenfeld, A., and Davis, L. S., "Random Mosaic Models for Textures", IEEE Transactions on Systems, Man, and Cybernetics, Vol. 8, Sept. 1978, pp. 694-702.

Q.33 Sinofsky, H. J., "Computer Color Halftone Generation Through Hardware/Software", in Proceedings of the 11(th) Annual Conference on Modeling and Simulation, Instrumentation Society of America, Research Triangle Park, North Carolina, May, 1980, pp. 147-153.

Q.34 Sutherland, I., Sproull, R. F., and Schumacker, R. A., "A Characterization of Ten Hidden-Surface Algorithms", Computing Surveys, Vol 8, March 1974, pp. 1-55.

Q.35 Switzer, P., Geometrical Measures for the Smoothness of Random Functions, Tech Report No. 62, Stanford University Statistics Department, 1974.

Q.36 Watkins, G. S., A Real-time Visible Surface Algorithm, University of Utah Computer Science Department, UTECH-CSC-70-101, June, 1970.

P- PERCEPTUAL PSYCHOLOGY IN VISUAL TRAINING

P.1 Beamon, W. S., Observer Performance During a Video Display, Proceedings of the Human Factors Society, 21st Annual Meeting, 1977, pp. 406-410.

P.2 Biberman, L. M., (Ed), Perception of Displayed Information, Plenum Press, New York, 1973.

P.3 Bunker, W. M., "Training Effectiveness Versus Simulation Realism", Proceedings of the 11(th) Industry Conference, NAVTRAEQUIPCEN IH-306, Naval Training Equipment Center, Orlando, Florida, Nov. 1978.

P.4 Bynum, J. A., Evaluation of the Singer Night Visual System Computer Generated Image Display Attached to the UH-1 Flight Simulator, U. S. Army Institute for the Behavioral and Social Sciences, Airfield Unit at Fort Rucker, Alabama, Research Report 1199, September, 1978.

P.5 Condron, T. P., Handbook of Geophysics, Chapter 14, Macmillan, New York, 1961.

P.6 Erickson, R. A., Image Identification of Television, Report NWC TP5025, Naval Weapons Center, China Lake, California, September, 1970.

P.7 Erickson, R. A., Line Criteria in Target Acquisition with Electro-Optical Devices, Naval Weapons Center, China Lake, California, Report No. NWC TP5854, March, 1976.

P.8 Erickson, R. A., and Hemingway, J. C., Image Identification of Television, Naval Weapons Center, Report No. NWC TP5025, China Lake, California, September 1970.

P.9 Erickson, R. A., and Hemingway, J. C., Relative Effects of Raster Scan Lines and Image Subtense on Vehicle Identification of Television, Report No. NWC IDP 2975, Naval Weapons Center, China Lake, California, August 1969.

P.10 Erickson, R. A., and Main, R. E., Target Acquisition on Television, Report No. NOTS TP 4077, Naval Ordinance Test Station, China Lake, California, August, 1966.

P.11 Hemingway, J. C., and Erickson, R. A., Effects of Raster Scan Lines and Image Subtense on Observer Performance on Television, Report No. NWC IDP 2883, Naval Weapons Center, China Lake, California, April, 1968.

P. 12 Image Modelling Conference, Williams AFB, December 1977, pp. 257-269.

P. 13 Kraft, C. L., Anderson, C. D., and Elworth, C. L., Psychophysical Criteria for Visual Simulation Systems, Boeing Aerospace Study for AFHRL, AFHRL-TR-79-30, May 1980.

P. 14 Meshier, C. W., and Butler, G. J., Air Combat Training in a Simulator, AFARF-CP-198, October, 1975.

P. 15 Richards, W., Experiments in Texture Perception, MIT DARPA report for AFOSR, AFOSR-TR-78-1165, January, 1978.

P. 16 Roscoe, S. N., "Visual Cue Requirements in Imaging Displays", in Proceedings of Image Modelling Conference held at Williams AFB, Arizona, December 1977, pp. 256-269.

P. 17 Sanders, N. G., Simmons, R. R., and Hoffman, M. A., Visual Workload of the Copilot/Navigator During Flight, U. S. Army Aeromedical Research Lab., Fort Rucker, Alabama, Dec. 1977.

P. 18 Schumacker, R. A., Brand, B., Gilliland, M., and Sharp, W., Study for Applying Computer Generated Images to Visual Simulation, AFHRL-TR-69-14, U. S. Air Force Human Resources Lab, September, 1969.

P. 19 Schumacker, R. A., and Rougelot, R. S., Image Quality: A Comparison of Night/Dusk and Day/Night CGI systems, Image Modelling Conference, Williams AFB, December, 1977, 257-269.

P. 20 Schwank, J. C. et al, Pilot Performance During Flight Simulation With Peripherally Presented Visual Cues, Proceedings of the Human Factors Society, 22nd Annual Meeting, 1978, pp. 222-225.

P. 21 Stark, E. A., "Motion Perception and Terrain Visual Cues in Air Combat Simulation", Proceedings of the AIAA Visual Motion and Simulation Conference, Dayton, Ohio, April 1976, pp. 39-49.

P. 22 Tamura, H., Shunji, M., and Yamawaki, T., "Textural Features Corresponding to Visual Perception", IEEE Transactions on Systems, Man, and Cybernetics, Vol. 8, June 1978, pp. 460-473.

P. 23 Tilton, H. B., "How Many Colors", Optical Spectra, February 1979, pp. 32-34.

P. 24 Welp, D. W., LaMuth, H. L., and Kuhner, M. B., Target Acquisition and Model Development, Battelle Columbus Labs, Camouflage Technology Center, Columbus, Ohio, September, 1976.

P. 25 Winkler, G., and Vattrodt, K., "Measures of Conspicuousness", Computer Graphics and Image Processing, Vol 8, 1978, pp. 355-368

V- ARITIFICIAL VISION AND IMAGE PROCESSING

V.1 Ahuja, N., Mosaic Models for Image Analysis and Synthesis, University of Maryland Computer Science Dept, Tech Report 607, Nov 1977.

V.2 Akin, O., and Reddy, R., "Knowledge Acquisition for Image Understanding Research", Computer Graphics and Image Processing, vol 6, August, 1977, pp. 307-334.

V.3 Arbib, M. A. and Riseman, E. M., Computational Techniques in Visual Scenes, Univ. of Mass. Computer and Information Science TR 76-10, TR 76-11, Amherst, Mass., 1976.

V.4 Ballard, D. H., Brown, C. M., and Feldman, J. A., An Approach to Knowledge Directed Image Analysis, University of Rochester, in Computer Vision Systems", A. R. Hanson and E. M. Riseman, (Eds.), Academic Press, 1978, pp. 271-281.

V.5 Binford, T. O., "Spatial Representation", in Proceedings of the Image Understanding Workshop in Cambridge Mass, May 1978, L. S. Bauman, (Ed.), Scientific Applications Inc, SAI-79-749-WA, Arlington, Va, 1979, pp. 109-111.

V.6 Brooks, R. A., Greiner, R., and Binford, T. O., "A Model Based Vision System", in Proceedings of the Image Understanding Workshop in Cambridge Mass, May 1978, L. S. Bauman, (Ed.), Scientific Applications Inc, SAI-79-749-WA, Arlington, Va, 1979.

V.7 Bullock, B. L., Dudani, S., Stafsudd, J., and Clark, C., Finding Structures in Outdoor Scenes, Research Report 498, Hughes Research Lab, Malibu, California.

V.8 Davis, L., Johns, S., and Aggarwal, J. K., Texture Analysis Using Generalized Cooccurrence Matrices, University of Texas Computer Science Department.

V.9 Ehrich, R. W., and Foith, J. P., "Topology and Semantics of Intensity Arrays", in Computer Vision Systems", A. R. Hanson and E. M. Riseman, (Eds.), Academic Press, 1978, pp. 111-127.

V.10 Fischler, M. A., "On the representation of natural scenes", in Computer Vision Systems", A. R. Hanson and E. M. Riseman, (Eds.), Academic Press, 1978, pp. 47-52.

V.11 Galloway, N. M., Texture Analysis Using Gray Level Run Lengths, Computer Graphics and Image Processing, Vol. 4, June 1975, pp. 172-179.

V. 12 Gonzalez, R. C., and Wintz, Digital Image Processing, Addison-Wesley, 1977.

V. 13 Hall, E. L., Computer Image Processing and Recognitinn, Academic Press, 1979.

V. 14 Haralick, R. M., Shanmugan, K., and Dinstein, I., Textural Features for Image Classification, IEEE Transactions on Systems, Man, and Cybernetics, Vol. 3, 1973, pp. 610-621.

V. 15 LeBente, A. E., "Two-Dimensional Image Coding by Micro-Adaptive Picture Sequencing (MAPS)", Proceedings of the SPIE, Vol. 119, Appication of Digital Image Processing, 1977, 99-106.

V. 16 LeSchack, L. A., Brinley, W. R., and McIvor, D. E., Automatic Processing of Airborne Remote Sensing Data for Pattern Discrimination of Jungle and Other Vegetative Areas, Dev. and Resources Transp. Corp., 1111 Univ. Blvd., Silver Springs, Md., May, 1973.

V. 17 Longuet-Higgins, M. S., "On the Statistical Distribution of the Heights of Sea Waves", Journal of Marine Research, Vol XI, 1952, pp. 245-266.

V. 18 Matern, B., "Spatial Variation", Medd Statens Skogsforsknings Institut, Stockholm, Sweden, 1960, pp. 1-144.

V. 19 Marr, D., "Representing Visual Information - A Computational Approach", in Lectures on Mathematics in the Life Sciences, Vol. 10, American Mathematical Society, 1978, pp. 61-80.

V 20 Nagao, M., Matsuyama, T., and Ikeda, Y., "Region extraction and Shape Analysis in Aerial Photographs", Computer Graphics and Image Processing, Vol 10, 1979, pp. 195-223.

V. 21 Nevatia, R., and Price, K. E., "Symbolic Representation", in Proceedings of the Image Understanding Workshop in Cambridge Mass, May 1978, L. S. Bauman, (Ed.), Scientific Applications Inc, SAI-79-749-WA, Arlington, Va, 1979. pp. 145-148.

V. 22 Pielou, E., Mathematical Ecology, Wiley, 1977.

V. 23 Rosenfeld, A., Picture Processing by Computer, Academic Press, New York, 1969.

V. 24 Rosenfeld, A., "Picture Processing: 1974", Computer Graphics and Image Processing, Vol 4, pp. 133-155, 1975.

V. 25 Rosenfeld, A., and Kak, A. C., "Digital Picture Processing", Academic Press, New York, 1976.

V.26 Rosenfeld, A., and Troy, E. B., "Visual Texture Analysis", in Proceedings of the UMR Hervin J. Kelly Communications Conference, University of Missouri, Rolla, Missouri, October 1970, Section 10-1.

V.27 Schachter, B. J., Davis, L. S., and Rosenfeld, A., "Some Experiments in Image Segmentation by Clustering of Local Feature Values", Pattern Recognition, Vol. 11, pp. 19-28, 1979.

V.28 Shanmugam, K. S., Dickey, F. M., and Green, J. A., "An Optimal Frequency Domain Filter for Edge Detection in Digital Pictures", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-1, January, 1979.

V.29 Sloan, K. R., and Bajcsy, R., World Model Driven Recognition of Outdoor Scenes, University of Rochester, Computer Science Tech Report TR-40., Sept, 1979.

V.30 Wechsler, H., "Texture Analysis - A Survey", Signal Processing, Vol. 2, pp. 271-282, 1980.

V.31 Weiman, C. F. R., Scene Analysis: A Survey, ONR Report number NSO-9, prepared under ONR contract N00014-75-C-0571 at Courant Institute for Mathematical Sciences, New York University, 1976.

V.32 Winston, P., The Psychology of Computer Vision, McGraw-Hill, New York, 1975.

V.33 Wong, E., "Two Dimensional Random Fields and Representation of Images", SIAM Journal of Applied Mathematics, Vol 16, 1968, pp. 756-770.

III. REVIEW OF KEY REFERENCES

Conduct of the background investigation and technology review activities defined by Study Tasks 1 and 2, respectively, involved an extensive literature search. The result of this search was the Topical Bibliography provided as Reference II. While much of the reference material which was directly applicable to this study has been incorporated into the main body of this report, there is other material of general interest which merits comment. Accordingly, some of this material has been paraphased to formulate this brief commentary. It should be understood that the annotations are intended to reflect an objective synopsis by the author of this report as to the relevant content of the material reviewed.

The key references are grouped into the following categories or subjects:

Code	Subject
C -	CIG and Visual Simulation.
F -	FLIR, LLLTV, Microwave, and other Visual Sensor Displays.
G -	Computer Graphics.
P -	Perceptual Psychology in Visual Training.
V -	Artificial Vision and Image Processing.

C - CIG AND VISUAL SIMULATION

C.4 P. K. Doenges, "Introduction to Computer Image Generation Technology for Visual Simulation," University of Dayton/AIAA Flight Simulation Short Course, Dayton, Ohio, March, 1980.

Raster scan devices can accommodate higher scene complexity through image generator parallelism, but cannot match calligraphic image quality. Calligraphic systems have limited scene complexity due to drawing rate limitations. While hybrid formats offer desirable characteristics of both types, display and projection technology for hybrid systems has lagged raster scan technology.

CIG system functionality during realtime scene processing is dominated by a very large data reduction (culling) operation. The system must pare down all the data describing a scene from all possible viewing angles into a highly restricted viewing set describing the instantaneous images of just those scene components within the field of view (FOV) visible near the observer.

A pancake window provides a very large circular FOV (about 90^{o}). It can be used either as a pupil forming reflective system or a non-pupil forming infinity display system. If the collimating eyepiece mirror is fed by an optical projection element directly such that an aerial image is presented to the mirror, then a rear pupil will be formed whose size will depend primarily on the F-number bundle of the projection optics. If the in-line system is fed by a screen, the infinity display will not form a rear pupil. When a rear pupil is not formed, the maximum FOV for instantaneous "best" viewing will occupy a particular volume at a particular distance from the window.

Beam penetration CRT's exploit the progressive penetration of electrons into materials as a function of their increased kinetic energy. The primary advantage of the beam penetration tube over the shadow mask tube lies in its continuous phosphor structure which affords it a high resolution capability approaching that of monochromatic tubes. The CRT also provides a limited color capability with a single electron gun, eliminating the need for color registration.

The General Electric light valve consists of a single electron gun used to write appropriate diffraction gratings on a transparent oil film. A slot and bar system in conjunction with a light source permits the desired color to be selected from the spectrum developed by each diffraction grating. Since the system uses an electron gun to produce a color image, no registration of the red, green, and blue colors is needed.

C.5 A. B. Doty and T. W. Hoog, "Components of a Simulator," In-course Notes, University of Dayton/AIAA Flight Simulation Short Course, Dayton, Ohio, March, 1980.

Sensors are used to provide ground map references for navigation, to aid the operator in locating, identifying, and tracking other airborne vehicles and ground targets, and in close terrain following. A terrain avoidance radar display may take the form of a trace showing the highest elevation radar return in front of the aircraft or as display of radar returns which exceed a selectable elevation above the ground. A terrain following radar is used by the pilot to fly at a specified flight path.

Sensors are used for ground mapping purposes in many ways. The term "ground mapping" is used because the sensor display provides a picture of the ground with recognizable features as would be found on a topographic map. Mapping sensors operate in both side looking and forward looking modes.

Ground mapping radar simulators have used three major methods of data storage over the past 25 years. The first was a 3-D model of the selected geographic area. This model was in a water tank which was used to simulate some radar characteristics. The second used monochrome or tricolor film transparencies. These transparencies

were scanned by a flying spot scanner to retrieve the encoded topographic data, and the signals were then processed in analog circuitry. Data such as terrain elevation is highly compressed and must be reconstructed to provide continuous terrain elevation values for the radar processor.

Sensor models should consider:

(a) atmospheric attenuation
(b) particle content of the atmosphere
(c) sun position
(d) time of day, time of year

(Note: For radar, time of day and sun position are not important)

Radar models should include:

(a) antenna scan rate
(b) antenna pattern
(c) radiated power
(d) attenuation due to aspect
(e) antenna tilt
(f) gain controls

Electro-optical sensor models should consider:

(a) scan rate
(b) scan type
(c) detector sensitivity
(d) detector sampling
(e) gain controls

D - GRID DATA BASE DISPLAY AND GENERATION

D. 14 McKeown, "Representation for Image Data Bases", in Proceedings of the Image Understanding Workshop, Cambridge, Massachusetts, May, 1978, L. S. Bauman, (Ed.) Scientific Applications, Inc., SAI-79-749-WA, Arlington, VA, 1979, pp. 109-111.

This paper covers concepts for the storage of terrain and culture in an integrated data base. A hierarchy of signal and symbolic descriptions is suggested. The authors conclude that terrain is best represented by surface-specific points, and culture by overlays.

F - FLIR, LLLTV, MICROWAVE, AND OTHER VISUAL SENSOR DISPLAYS

F. 1 J. W. Bell, I. P. Bottik, and A. B. Lucero, Simulation Techniques for Airborne Electro-Optical Imaging Systems, Technology Services Corporation, AFHRL-TR-75-43, U. S. Air Force Systems Command, Brooks AFB, Texas, December, 1975.

This report discusses the non-realtime simulation of FLIR and LLLTV sensors using a general-purpose computer. The data base is divided as follows:

(1) Attenuation coefficients
(2) Target Geometry
(3) Intensity levels
(4) Random background features

Attenuation Coefficients

Attenuation is caused by two mechanisms: absorption and scattering. Absorption is caused by water vapor and carbon dioxide in the atmosphere. Scattering is caused by haze.

Target Geometry

Polygons are defined by strings of vertices, circles by center points and radii. Data bases are constructed from photos and blueprints with the aid of a digitizing tablet.

Intensity Levels

For LLLTV, all intensity levels reflected by an object under shade were estimated to be one-fourth that of the object in sunlight. FLIR intensity levels were estimated. Shadows were computed by projecting an object's vertices onto the ground plane.

F.4 M. Bunker and R. Heeschen, Airborne Electro-optical Sensor Simulation, AFHRL, Brooks AFB, Texas, July, 1975.

This report discusses the simulation of sensor displays with a modified version of a General Electric flight simulator visual system.

F.6 S. de Groot, Human Factors Aspects of LLLTV and FLIR Sensor Displays, Air Force Office of Scientific Research, Air Force Systems Command, Bolling AFB, D. C., AFOSRTR 78-1237, 1978.

The objectives of this study were to:

(1) Scale subjectively the complexity of still FLIR and LLLTV scenes.
(2) Identify scenes of differing complexity for comparison and future simulation.
(3) Determine empirically the major perceptual factors associated with FLIR and LLLTV.
(4) Relate perceptual physical factors amenable to CIG simulation.

Scene complexity is important because it is related to both realism and the cost of generating the image.

For low-level flight under FLIR and LLLTV simulation, difficulty in detecting heights could be critical. At those times of day when FLIR and LLLTV displays closely approximate daylight scenes, texturing would undoubtedly help in terrain avoidance and terrain following.

Evaluation of LLLTV Scene Complexity: Most subjects tested viewed background terrain, landscape, foliage, etc., as unimportant as compared to man-made objects. Some subjects related the information content of the images to the number of man-made objects appearing in the scene. The physical parameter represents a mere numeric counting following an explicit evaluation of what constitutes a man-made object. The parameters for measuring scene complexity, in order of importance; were defined to be:

(1) Number of man-made objects in scene.
(2) Ratio of man-made objects to background (sky, foliage, terrain, etc.).
(3) Visible edge count for objects.
(4) Scene quality as reflected in resolvable detail.

Different man-made objects were found to have differing importance. Objects of low importance included scattered houses, apartments, churches, microwave and water towers, and overpasses. Objects of high importance included hospitals, grain elevators, aircraft hangers, and factories.

Evaluation of FLIR Scene Complexity: Most subjects considered the scaling of FLIR photos in terms of scene complexity not only harder than for LLLTV, but a different kind of task. The background in FLIR photos was considered more important than for LLLTV. This may be because subjects often had difficulty distinguishing objects from the background. Some subjects related information content to object detail.

The lack of correspondence between the two analyses indicates that visual parameters and synthesized measures adequate for the quantification of LLLTV may not be entirely suited for FLIR imagery. In FLIR displays subjects placed a high importance on patterns of light areas and gave a lower importance to the shape of objects.

Under some weather and lighting conditions, FLIR and LLLTV can produce images that are very similar to normal visual images. At other times, FLIR and LLLTV will produce displays with different characteristics, but different in a predictable manner. For example, FLIR sensors are particularly sensitive to the relative changes occuring around sunset as the heating properties of the sun are withdrawn, and natural patterns of heat emissivity of objects are registered. LLLTV sensors are notoriously oversensitive to some light sources in the night environment - displaying the phenomena of blooming. Blooming refers to small lights enlarging to flood the entire display.

Descriptors Used To Describe LLLTV Scenes:

Number of objects
Background
Object detail: shape
Proportion of figure to ground
Object detail: substructure interior complexity
Resolution
Edge count

Descriptors Used To Describe FLIR Scenes:

Number of objects
Edge gradient
White = hot = man-made
Proportion of figure to ground
Missing detail, masked, washed out
Background
Patterns

F.7 H. Dessau, AWS Image Simulation Study, U. S. Air Force Rome Air Development Center, TR-78-176, August, 1978.

The objective of this study was to define the essential parameters of various sensors that are required to support synthetic sensor predictions. The parameters thus defined were to be transformed into detail data base descriptors.

LLLTV operates in a spectral bandwidth in which each photon contains more energy than FLIR. LLLTV's are capable of operating by faint starlight.

The author concludes that as long as the signal to noise ratio is at an acceptable level (such as 1.2), then the number of scan lines should be chosen so that the smallest target feature subtends an angle no smaller than 5 arc minutes.

The one area in which FLIR has a theoretical advantage over LLLTV is the atmosphere effects of attenuation by absorption and scattering.

Poorly insulated houses conduct heat better to the roof than well insulated ones. Unused buildings are usually cool.

LLLTV and FLIR can be used to distinguish different types of vegetation (radar is not so effective for this purpose). Crop patterns should be modelled as textures having statistical characteristics which depend on season and climate. During rain, FLIR images have a washed-out look.

The author suggests that point targets should be modelled by center-satellite clustering processes, and that terrain be modelled with Gaussian random fields and man-made patterns with line mosaics.

F.10 E. D. Moldonato, Forward Looking Infrared Simulation, Naval Training Equipment Center, NAVTRAEQUIPCEN TN-39, Orlando, Florida, July, 1974.

One of the most important uses of FLIR imagery is for reconnaissance. Trainees must be taught search, target detection, target acquisition, and identification functions.

In the area of navigation, trainees must be taught to recognize landmarks and correlate map readings with a FLIR display. It is possible that not all landmarks or terrain features will appear in the FLIR display, or that their character will be changed by the infrared radiance of the feature.

FLIR systems are used as aids in takeoffs and landings when there is mist or ground fog.

F.13 N. L. Schult and F. Wenthen, Advance Weapon System Image Simulation II Study Final Report, U. S. Air Force Rome Air Development Center, RADC-TR-80-185, June, 1980.

A theoretical approach to the modelling of FLIR scenes is used. A scene is viewed as an arrangement of features, a feature as an arrangement of objects, and an object as an arrangement of surfaces. Each surface has a generic material type. A typical scene feature is a building. The roof of the building may, for example, be modelled as layers of paint, steel, pine, plaster, felt, pine, air, and plasterboard. Material characteristics are specified in terms of such properties as conductivity, specific heat, density, absorptivity, and emissivity. Time-variant thermal properties are evaluated as a function of latitude, longitude, time of day, time of year, etc. Equations describing the thermal properties of each surface within the scene are formulated and solved. These surface temperatures are then used in a simulation of perspective views of scenes.

F.16 A. D. Stathacopoulos and H. F. Gilmore, Selection of Mathematical Models of Target Acquisition by Electro-opitcal Ssystems, Science and Technology Corporation, Santa Barbara, California, June, 1977. (Naval Weapons Center Technical Publication 5928.)

A review of fourteen sensor models is presented. All the models reviewed represent the displayed scene to which the observer responds by a combination of some of the following descriptors:

(1) Target coordinates on the display.
(2) Target angular subend at the observer's eye.
(3) Target/background contrast.
(4) Displayed target and background luminance.
(5) Resolution of the electro-optical system.
(6) Target dimension on the display.
(7) Displayed 2-D noise.
(8) Background structures and clutter.
(9) Display size.
(10) Eye integration and fixation periods.
(11) Search time available to observer.

Of these, the target-related and time factors are either input parameters or are determined by straightforward calculations. Target and background signal levels are determined by the spectral reflectance and emittance of the target and the background, and are affected by the atmosphere (absorption and scattering) and the transfer function of the electro-optical system. The resolution of the system is usually expressed in the number of TV lines.

Target Background Representation

In all sensor models, the descriptors for targets and backgrounds include only the most basic parameters. Targets are usually represented by rectilinear blocks, or are approximated by periodic patterns (e.g., squares, circles) or bar targets. For LLLTV it is usual to specify target reflectance and luminance. For FLIR, emissivity and temperature differences with respect to the background are specified. It is always assumed that targets are viewed against a uniform background. All of the more subtle characteristics of optical signatures such as shadows, perspective, and highlights are not modelled. Shadows are sometimes considered.

Atmospheric Effects

Molecular absorption is of primary importance in the IR portion of the spectrum and of secondary importance in the visible region. All the electro-optical sensor models use certain approximations to estimate the absorption suffered by radiation in traversing

a given amount of water vapor. Only some models consider absorption by CO_2. These approximations give average values of transmissivity over a given wavelength region; thus, the models are limited to illumination and emission over relatively broad spectral regions. There are significant differences in the sophistication with which different models compute approximate values of absorption. Most models are based on the methods published by Altshuler, Larmore, and Kruse.

Aerosol scattering is usually the limiting mechanism for propagation of visible radiation, except in extremely clear atmosphere. For IR radiation, aerosol scattering is of importance in thick haze, fog, or smoke. Not all models include the effect of aerosol scattering; those that do use simple formulations applicable only to haze. These formulations are based on the work by Langer and Rensch which involves the calculation of the extinction coefficients from the visibility range, which is provided as an input.

None of the models treats aerosol absorption or the effects of atmospheric turbulence. These are secondary effects, but can be important under specific atmospheric conditions.

Sensor Representation

Representation of the electro-optical system involves mostly relatively well-defined concepts of physics and engineering. The mathematical tools are those of linear system theory and Fourier analysis. Practically all models use this basic approach. Computations regarding the spectral distribution of radiation emitted or reflected by the target, the spectral transmission of the atmosphere, and the response of the sensor are usually treated in simplified formulation.

The performance of TV camera tubes is usually specified by providing the following relationships either in the form of curves or tabulation: TV lines resolved per raster height as a function of cathode illuminance, with contrast modulation as a parameter; output current as a function of photo-cathode illuminance; and signal to noise ratio as a function of photo-cathode illuminance.

In IR systems, the characteristics of different components are specified in terms such as detector specific detectivity, amplifier bandwidth, and collector area; or, the overall performance is described by giving its noise equivalent temperature, instantaneous FOV, system modulation transfer function, etc.

The determination of SNR is usually straightforward. For TV camera tubes, it is obtained from the computed value of cathode illuminance and from the manufacturer's specification of SNR for that illuminance. For FLIR, it is the usual practice to compare the collected target-background power difference with the detector noise equivalent power.

Resolution for TV is obtained from curves relating the number of lines resolved per picture height to photosurface illuminance and contrast modulation. For FLIR, a common simplification is to use the angle subtended by the detector to specify resolution; the modulation transfer function is sometimes evaluated to obtain a measure of the number of lines resolved per picture height. Contrast is obtained directly from the displayed target and background illuminance, or calculated from the actual target-background reflected or emitted radiation modified by the atmosphere and the stages of the system.

F.17 M. E. Tryon, G. L. VanDeman, C. E. Hogue, and W. J. Bridges, Computer Assisted Simulation of Electromagnetic Sensors, Naval Avionics Facility, TR 2021, September, 1975.

The purpose of this study was to develop computer simulation models for predicting the performance of active and passive, LLLTV, FLIR, and MICRAD sensors.

G - COMPUTER GRAPHICS

G.1 A. Baer, C. Eastman, and H. Henrion, A Survey of Geometric Modelling, Carnegie-Mellon University Institute of Physical Planning, Report No. 66, March, 1977.

Ten computer models for three-dimensional objects are surveyed. The majority of the models reviewed are oriented toward mechanical engineering or architecture. Geometric objects may be described in terms of surfaces, edges, or vertices. A vertex description is generally employed when the object is to be displayed calligraphically. Surface normals (or the equivalent face coefficients) are used with a raster scan display approach. Some systems store all three types of geometrical information; others store a minimal representation and compute the rest when needed. Some systems represent curved surfaces as generic solids (e.g., cylinders, cones) or surface patches. The curved edges bounding a surface are sometimes represented as cubic splines. A polyhedron's topology may be specified in terms of any of the nine possible relations among faces, edges, and vertices. The choice of a representation involves space and time tradeoffs.

G.12 C. M. Eastman, "The Concise Structuring of Geometric Data for Computer Aided Design," in Data Structures, E. A. Klinger and K. S. Fu and T. L. Kunii, Academic Press, 1977, 31-58.

Any shape may be considered to be a polyhedron made up of vertices, edges, and faces. The faces of a polyhedron may be planar or curved. The complete description consists of a TOPOLOGY of the adjacency relations between its constituent parts, and a GEOMETRY specifying the dimensional aspects of the shape. Without both aspects,

a shape description is incomplete and some operations on it are not possible. In most data structures, these two aspects are not separated from each other.

A commonly used form of redundancy in graphical data bases is that of whole shapes. The same part or shap may be used in many different locations. Each need not be defined separately. Instead, a spatial transformation can be used to define the location of each instance.

The compute-vs-store question is essentially an optimization problem seeking the best combination of storage and computing costs.

G. 13 R. L. Elliot (of Los Alamos Laboratory), Computer Graphics in a Multiple System, Multiple Device Environment, in Proceedings of Computer Graphics Conference, Brighton, England, August 13-15, 1980, 99-110.

A system composed of four types of computers which can talk to each other plus a number of graphic output devices is covered. The routines used generate graphic commands in a common format for all graphic device drivers. Only those drivers needed at any moment are linked and loaded. World coordinates are automatically transformed into normalized device coordinates for the device addressed.

This approach is based upon the use of a metafile, which is a bit stream with no record marks or other system type data embedded in it. It serves as a vehicle for transfer between the different computer's operating system; no conversion is required. The metafile consists of file descriptor commands followed by graphic commands. A file descriptor command consists of an ASCII character string, giving the attributes of the metafile. The metafile serves as an input to graphic output devices. Device and media dependent parameters are all contained in the file descriptor command, making it trivial for the graphic service utilities to direct the output. If several views of the same scene are to be displayed, separate workspaces are maintained for each view.

Graphic service utilities are used to direct the metafile to a particular medium. Graphic editors are used to modify the metafiles. Graphic generators are used to combine text with scenes.

G. 17 J. Frazer, J. Frazer, and P. Frazer, "Intelligent Three-Dimensional Modeling Systems", in Proceedings of Computer Graphics Conference, Brighton, England, August 13-15, 1980, 359-370.

An interesting approach to inputting 3-D model data into the computer by means of a model board and a set of plug-in physical models is described.

The data base modeller simply plugs the models into the board in any arrangement he desires, and the computer reads the locations and model types from the board. This approach can also be used for playing chess with the computer.

G. 22 R. L. Henderson, Geometric Reference Preparation Interim Report No. 1: The Broken Segment Matcher, Control Data (for RADC), Minneapolis, MN, April, 1979. Geometric Surface Shell Studies: Final Report, Control Data, August, 1979.

These reports demonstrate the feasibility of automatically modelling three-dimensional culture targets composed of planes. Stereo pairs are used in the automatic fitting of planar surfaces to real world objects, such as buildings. The result is a "block world" model of a city.

The central technical problem that is attempted to be solved is the accommodation of surface discontinuities. The problem of dealing with curved surfaces remains untouched. The software described is now structured to look for sharp changes in slope and occlusion. The difficulties associated with rolling terrain are ignored.

Conclusions drawn from these reports are that the goal of total automation is still far from fulfilled, concepts for automatic curved surface modelling have yet to be developed, and the substructure modelling problem has not yet been addressed. Their approach has only been tested on a very limited data set.

G. 30 D. M. Russell, Constraint Networks, University of Rochester C. S. T. R. 38, August, 1978.

A "constraint network" models the location of real world objects in terms of their relationships to other objects already located in the image. The principal use of this model is to search for features in an image.

Other information may be built into a constraint network. For example, it may be known that docks have an albedo within a certain range. Knowing this fact reduces the search time for finding a dock.

The constraint network could be given a task such as finding the most likely place that an enemy armored vehicle (tank) will hide.

P - PERCEPTUAL PSYCHOLOGY IN VISUAL TRAINING

P. 1 W. S. Beamon, "Observer Performance Using a Video Display," Proceedings of the Human Factors Society, 21st Annual Meeting, 1977, 406-410.

Since FLIR and LLLTV sensors are usually operated under poor radiance conditions, factors affecting the quality of the displayed image are deserving of close scrutiny.

The author suggests reducing the modulation of the raster structure by employing spot wobble. This results in a significant increase in the ground range at which targets are detected if the video signal is relatively noise-free.

P.4 J. A. Bynum, Evaluation of the Singer Night Visual System Computer Generated Image Display Attached to the UH-1 Flight Simulator, U. S. Army Institute for the Behavorial and Social Sciences, Airfield Unit, Fort Rucker, AL, Research Report 1199, September, 1978.

The conclusion of the author was that there was no evidence of any increase in skill as a function of simulator training. On the basis of the data collected, the tested device was judged not suitable for training night contact helicopter maneuvers.

Recommendations:

(1) Improve cockpit environment - eliminate light leaks.
(2) Adjust eye height to give perceptually corrective perspective and focus.
(3) Improve surface texture.
(4) Improve scene luminance.

P.14 C. W. Meshier and G. J. Butler, Air Combat Training in a Simulator, AFARF-CP-198, October, 1975.

This is an informal discussion of the effectiveness of simulators for pilot training.

P.16 S. N. Roscoe, Visual Cue Requirements in Imaging Displays, Image Modelling Conference, Williams AFB, AZ, December, 1977, 257-269.

Very crude computer generated line drawings of runways were used to investigate what visual parameters were important during landing. Some obvious conclusions were drawn; for example, runway texture taking on added importance as the runway is neared.

P.17 N. C. Sanders, R. R. Simons, and M. A. Hoffman, Visual Workload of the Copilot/Navigator During Flight, U. S. Army Aeromedical Research Laboratory, Fort Rucker, AL, December, 1977.

Three basic flight profiles are considered:

Nap of the Earth (NOE) - Fly as close to the earth as vegetation or obstacles will permit. Airspeed and altitude are varied as influenced by the terrain, weather, and enemy situation. The pilot preplans a broad corridor of operation based upon terrain, weather, and enemy situation.

Contour - Fly at a low altitude conforming to the general contours of the earth. This type of flight takes advantage of available cover and concealment to avoid detection. It is characterized by varying airspeed and obstacle avoidance.

Low Level - Fly at an altitude selected to minimize the possibility of detection. The route is preselected and conforms generally to a straight line, with constant airspeed and altitude.

The pilot's primary responsibility is maintaining clearance of the aircraft from all obstacles, and directing the plane over the desired route. The copilot (1) monitors the map and navigation instruments as well as the terrain in an attempt to locate the significant navigation cues needed for maintaining the correct flight path, (2) monitors other instruments, (3) tunes radios, (4) orally provides data to pilot, and (5) helps the pilot locate and avoid obstacles.

The frequency of eye fixation on any instrument is an indication of the relative importance of the instrument. The length of fixation indicates the difficulty of interpreting the instrument.

When flying below 100 feet, the avoidance of 3-D objects becomes primary.

The primary act of navigation in a rotary wing NOE, low level or contour environment, could be described as one of pattern comparison between the map and the terrain in sight. However, before pattern matching can occur, the navigator must first perform a search task for critical geographic features. Navigation requires the constant integration of information deemed critical on the map and its comparison to features visually extracted from the actual terrain. The navigator's task is made difficult by the fact that he must (1) view the terrain in a variety of states, e.g., seasonal changes, differing visibility and illumination conditions, etc., and (2) compensate for discrepancies between the map and the terrain in areas where significant terrain features have been changed (e.g., fields cleared, roads and bridges added, etc.).

This report provides charts delineating the visual workload of the pilot and navigator.

P. 19 R. A. Schumacker and R. S. Rougelot, Image Quality: A Comparison of Night/Dusk and Day/Night CGI Systems, Image Modelling Conference, Williams AFB, AZ, December, 1977, 243-255.

This paper discusses aliasing and similar image degradation problems.

P. 20 J. C. Schwank, et al., "Pilot Performance During Flight Simulation with Peripherally Presented Visual Cues," Proceedings of the Human Factors Society, 22nd Annual Meeting, 1978, 222-225.

Peripheral motion cues are effective orientation indicators. This paper discusses new methods for providing instrument readouts to the pilot.

P. 23 H. B. Tilton, "How Many Colors?", Optical Spectra, February 1979, 32-34.

The author divides the visible spectra into fifteen distinct colors.

P. 24 D. W. Welp, H. L. LaMuth, and M. B. Kuhner, Target Acquisition and Model Development, Battelle Columbus Laboratories, Camouflage Technology Center, Columbus, OH, September, 1976.

The objective of this effort was to further the development of camouflage analysis capabilities. A very good model development procedure is used.

Equations are developed for atmospheric transmission and scattering under a number of weather conditions. Appendices cover the probability of finding a target in a given time frame, color discrimination models, and FLIR models.

P. 25 G. Winkler and K. Vattrodt, "Measures of Conspicuousness," Computer Graphics and Image Processing Vol. 8, 1978, 355-368.

Conventional pattern recognition and correlation techniques for detecting/recognizing interesting objects in images use certain information about the objects. Knowledge of their shape, contour, and class membership is required before conventional methods of pattern recognition and detection can be applied. This article considers cases where either this information is not available or the object's characteristics are changing so fast that no interim updating is possible. Under these circumstances, a measure of conspicuousness is useful. This approach is viable for detecting objects in a field.

V. 2 D. Akin and R. Reddy, "Knowledge Acquisition for Image Understanding Research", Computer Graphics and Image Processing Vol. 6, August, 1977, 307-334.

Recent studies in cognitive psychology have accumulated considerable data on the information processing aspects of perception. Little research has been done on determining how a visual system decides which features are important, and how these features relate to the observer's knowledge. The usual measures used in perception are based upon reaction times and eye fixations. Most studies deal with abstract stimuli having a measurable information content.

In the experiments upon which this paper was based, the authors presented personnel with images to study, and later asked them questions about the content of the images without the images available for reference. The authors arrived at three stages for image understanding:

(1) OBTAIN VISUAL ATTRIBUTES OF SCENE COMPONENTS - Color, shape, location, orientation, and texture are considered important attributes.

(2) TRANSLATE VISUAL INFORMATION INTO AN UNDERSTANDING OF THE OBJECTS IN THE SCENE - Knowledge aids in translating individual features into meaningful objects.

(3) GENERATE CONCLUSIONS ABOUT THE SCENE

The levels of scene description that were referred to by the subjects are: scene, cluster, object, region, subregion, and segment. Features were described in terms of: location, size, shape, quantity, color, and texture.

V. 3 M. A. Arbib and E. M. Riseman, Computational Techniques in Visual Scenes, University of Massachusetts Computer and Information Science TR 76-10, 11, 1976, Amherst, Massachusetts.

The authors describe a system composed of two major elements: (1) a low level subsystem for segmenting an image into regions, and (2) a high level subsystem whose goal is the interpretation of an image by building a 3-D semantic representation of the world. The low level system is a hierarchical structural array of local processes which transform and reduce the image in parallel. The high level system employs modular processes which construct a model by interfacing the symbolic output of regions and boundaries to stored world knowledge in a semantic data base.

V. 4 D. H. Ballard, C. M. Brown, and J. A. Feldman, An Approach to knowledge - Directed Image Analysis, University of Rochester C. S. T. R. 21, 1977.

The authors define image analysis as a process of attaching meaning to an image. One way to do this is to use an explicit model of what the image can contain and then construct a mapping between the model and the image. A particular image is a specific realization of the model. Thus, the model must in some sense be more general than any given image. A useful model for a class of images will contain a large amount of information related to possible image content. However, the mapping generated by the analysis process will use only a small portion of this information.

The basic problem considered is one of querying the model to extract certain explicit information from it. One of the problems in generating the mapping is satisfying the query that the mapping is between very different structures. The natural elements of the image are pixels. To bridge this gap, the authors have structured their system in layers. At the most abstract end of the structure is a semantic network representing the model. The model contains generic information encoded as idealized prototypes of objects (which may be formed from edges).

At the intermediate level, they have a sketchmap. This is a data structure that is synthesized during image analysis and provides associations between the model and the image.

At the third layer is the image data structure. It consists of the original image at different magnifications, spectra, resolutions, etc., together with "filtered" versions of the image (texture image, edge image, etc.).

An example of an application is a problem in which a ship is to be found in an aerial photo. First, the photo is placed on top of a registered and scaled topographic map. The topographic map identifies the water areas (i.e., sea level). Then, the water areas in the photo are searched and the ship is located with a ship fitting template.

V. 6 R. A. Brooks, R. Greiner, and T. O. Binford, "A Model Based Vision System", in Proceedings of the Image Understanding Workshop, Cambridge, Massachusetts, May, 1978, (Editor) L. S. Bauman, published by Scientific Application, Inc., (SAI-79-749-WA), Arlington, Virginia.

The primary representation is in terms of three-dimensional generalized cones for volume elements. Their representation uses three principal graphs: a 3-D object graph, a 2-D appearance graph, and an observability graph which has both 2-D and 3-D features. The contents of the latter two of these graphs are in terms of the first, and may change over the course of recognition and display tasks. These latter two graphs always contain pointers to the object graph (and possibly to each other).

The 3-D object graph contains both generic and specific hierarchical models. At the highest level there are SCENES. SCENES are made up of OBJECTS and their spatial interrelationships. OBJECTS are graphs of attached PARTS. PARTS are subgraphs whose primitives are single generalized cones.

V. 10 M. A. Fischler, "On the Representation of Natural Scenes", Computer Vision Systems, Edited by A. R. Hanson and E. M. Riseman, 1978, Academic Press.

Visual perception is typically viewed as a partitioning of a scene into objects. Such primitive objects are characterized by some fixed set of attributes, and the scene is characterized by linking the objects together.

This paper questions the efficacy of the current approach used in machine vision; that is, (1) the symbolic/linguistic approach to scene representation and analysis, and (2) the use of a segmented scene upon which the analysis is based. This paper says that what is required is a device which can support a representation which is isomorphic to the spatial properties of the real world.

V.20 M. Nagao, T. Matsuyama, and Y. Ikeda, "Region Extraction and Shape Analysis in Aerial Photographs," Computer Graphics and Image Processing, Vol. 10, 1979, 195-223.

This paper contains interesting ideas for extracting three-dimensional objects from aerial photographs. Photographs of the same scene taken at two different times of day can be subtracted to obtain shadow areas. The shadows help locate 3-D objects and provide information concerning their size.

V. 21 R. Nevatia and K. E. Price, "Symbolic Representation", in Proceedings of the Image Understanding Workshop, Cambridge, Massachusetts, May 1978, L. S. Bauman, (Editor), Scientific Applications Inc., SAI-79-749-WA, Arlington, VA, 1979, pp. 145-148.

The authors develop semantic networks for objects and their relationships.

Objects can be generic or specific, simple or complex. Complex objects are represented by their parts and the relations among them. Linear features are represented by piecewise linear approximations, and roads by their medial axes.

V. 29 K. R. Sloan and R. Bajcsy, World Model Driven Recognition of Outdoor Scenes, University of Rochester, C. S. T. R. 40, September, 1979.

This report describes simple LISP programs for recognizing natural (non-manmade) objects in scenes. These programs are usually (but not always) able to find features such as the horizon in outdoor scenes.

MISSION
of
Rome Air Development Center

RADC plans and executes research, development, test and selected acquisition programs in support of Command, Control Communications and Intelligence (C^3I) activities. Technical and engineering support within areas of technical competence is provided to ESD Program Offices (POs) and other ESD elements. The principal technical mission areas are communications, electromagnetic guidance and control, surveillance of ground and aerospace objects, intelligence data collection and handling, information system technology, ionospheric propagation, solid state sciences, microwave physics and electronic reliability, maintainability and compatibility.